1. Write a C program to create a sequential file with at least five records, each records, each record having the structure shown below :

| USN | Name | Marks 1 | Marks 2 | Marks 3 |
|---|---|---|---|---|
| Non-zero Positive integer | 25 Characters | Positive integer | Positive integer | Positive integer |

Write necessary functions

a) To display all the records in the file.

b) To search for a specific record based on the USN. In case the required record is not found, suitable message should be displayed. Both the options in this case must be demonstrated.

**ALGORITHM FOR FILES PROGRAM.**

MAIN FUNCTION()

S1: Read the filename from user .
S2: Read the operations to be performed from the keyboard .
S3: If the operation specified is insert go to the insert function , if the operation specified is search go to the search function, if the operation specified is display go to the display function, if the operation specified is exit go to step 4.
S4: Stop.

INSERT FUNCTION()

S1: Open the file in the write mode ,if the file specified is not found or unable to open then display error message and go to step5 , else go to step2.
S2: Read the no: of records N to be inserted to the file .
S3: Repeat the step4 N number of times .
S4: Read the details of each student from the keyboard and write the same to the file .
S5: Close the file .
S6: Return to the main function .

SEARCH FUNCTION()

S1: Open the file in the read mode ,if the file specified is not found or unable to open then display error message and go to step6 , else go to step2.
S2: Read the USN number of the student whose details need to be searched .
S3: Read each student record from the file.
S4: Compare the students USN number scanned from file with USN number specified by the user.
S5: If they are equal then display the details of the searched record else display required USN number not found message and go to step6.
S6: Close the file.
S7: Return to the main function.

DISPLAY FUNCTION()

S1: Open the specified file in read mode.
S2: If the file is unable to open or not found then display error message and go to step4 else go to step3.
S3: Scan all the student details one by one from file and display the same at the console until end of file is reached.
S4: Close the file.
S5: Return to the main function .

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>

struct Student
{
        char name[25];
         int  usn;
         int m1,m2,m3;
};

char filename[15];
int n;
struct Student s;

void Insert()
{
        FILE *fp;
        int i;
        printf("\n\n\tEnter how many Records: ");
        scanf("%d",&n);
        fp=fopen(filename,"w");
        if(fp==NULL) printf("\n\n\tError in Creating the File....");
        else
                for(i=0;i<n;i++)
                {
                        printf("\nEnter Details of Student-%d: ",i+1);
```

```
                printf("\n\t\t\tName: "); scanf("%s",s.name); fprintf(fp,"%s
        ",s.name);
                printf("\n\t\t\tUsn : "); scanf("%d",&s.usn);  fprintf(fp,"%d
        ",s.usn);
                printf("\n\t\t\tM1  : "); scanf("%d",&s.m1); fprintf(fp,"%d ",s.m1);
                printf("\n\t\t\tM2  : "); scanf("%d",&s.m2); fprintf(fp,"%d ",s.m2);
                printf("\n\t\t\tM3  : "); scanf("%d",&s.m3); fprintf(fp,"%d ",s.m3);
        }
    fclose(fp);
}

void Search()
{
    if(n==0)
        printf("\n\n\tNo Records to Search...");
    else
    {
         FILE *fp;
         int i,flag=0,usn;
        fp=fopen(filename,"r");
        if(fp==NULL)
            printf("\n\n\tERROR!! in Opening the File....");
        else
        {
            printf("\n\n\tEnter the Usn to be Searched: ");
            scanf("%d",&usn);
            for(i=0;i<n;i++)
            {
                fscanf(fp,"%s",s.name);
                fscanf(fp,"%d",&s.usn);
                fscanf(fp,"%d",&s.m1);
                fscanf(fp,"%d",&s.m2);
                fscanf(fp,"%d",&s.m3);
                if(usn==s.usn)
                {
                    flag=1;
                     break;
                }
            }
            if(flag==1)
            {
                 printf("\nRecord Exist And Details are:");
                 printf("\n\t\t\tName: %s",s.name);
                 printf("\n\t\t\tUsn : %d",s.usn);
                 printf("\n\t\t\tM1  : %d",s.m1);
                 printf("\n\t\t\tM2  : %d",s.m2);
```

```
                                printf("\n\t\t\t\tM3  : %d",s.m3);
                        }
                        else
                                printf("\n\n\tRecord having Usn %d Does Not
Exist...\n\n",usn);
                }
                fclose(fp);
        }
}



void Display()
{
        if(n==0)
                printf("\n\n\tNo Records to Display....\n\n");
        else
        {
                int i;
                FILE *fp=fopen(filename,"r");
                if(fp==NULL)
                        printf("\n\n\tERROR!! in Opening File...");
                else
                        for(i=0;i<n;i++)
                        {
                                printf("\nDetails of Student-%d: ",i+1);
                                fscanf(fp,"%s",s.name); printf("\n\t\t\tName: %s",s.name);
                                fscanf(fp,"%d",&s.usn);  printf("\n\t\t\tUsn : %d",s.usn);
                                fscanf(fp,"%d",&s.m1); printf("\n\t\t\tM1  : %d",s.m1);
                                fscanf(fp,"%d",&s.m2); printf("\n\t\t\tM2  : %d",s.m2);
                                fscanf(fp,"%d",&s.m3); printf("\n\t\t\tM3  : %d",s.m3);
                         }
                fclose(fp);
        }
}


void main()
{
        int ch;
        clrscr();
        printf("\n\n\tEnter the File Name: ");
        scanf("%s",filename);
        while(1)
        {
```

```
                printf("\n\n\t1.Insert...\t2.Search...\t3.display...\t4.Exit...");
                printf("\n\n\tEnter Your Choice: ");
                scanf("%d",&ch);
                 switch(ch)
                 {
                        case 1: Insert();  break;
                        case 2: Search(); break;
                        case 3: Display(); break;
                        case 4: exit(0);
                        default: printf("\n\n\tEnter Proper Choice...");
                }
        }
}
```

**OUTPUT**

```
    Enter the File Name: test

    1.Insert...   2.Search...   3.display...   4.Exit...

    Enter Your Choice: 1

    Enter how many Records: 1

Enter Details of Student-1:
            Name: nikhil

            Usn : 100

            M1  : 23

            M2  : 24

            M3  : 25


    1.Insert...   2.Search...   3.display...   4.Exit...

    Enter Your Choice: 2


      Enter the Usn to be Searched: 100

Record Exist And Details are:
                Name: nikhil
```

Usn : 100
M1 : 23
M2 : 24
M3 : 25

1.Insert... 2.Search... 3.display... 4.Exit...

Enter Your Choice: 2

Enter the Usn to be Searched: 101

Record having Usn 101 Does Not Exist..

2. Write and demonstrate the following C functions:
a. newStrCpy that does the same job as strcpy
b. newStrCat that does the same job as strcat without using any library functions.

**ALGORITHM FOR STRING COPY AND STRING CONCATINATION**

MAIN FUNCTION()

S1: Read string (source string) called Str1 from the keyboard.
S2: Call the string copy function with Str1 as the input parameter.
S3: Read another string called Str2 from keyboard.
S4: Call string concatenate function with Str1 and Str2 as input parameters.
S5: Stop

STRING COPY FUNCTION()

S1: Consider an empty string called Dest and accept Str1 string from calling function.
S2: Consider two variables called i and j and initialize i to the first character of source string and j to the first free location of the string d.
S3: Repeat steps step4 through step5 until end of Str1 string is reached.
S4: Copy the value of i to the jth location of the destination string Dest.
S5: Increment value and assign next character of string Str1 to i.
S6: Display string Dest. [copied string].
S7: Return to the main function .

STRING CONCATENATE FUNCTION()

S1: Consider an empty string called Dest and accept Str1 and Str2 strings from the calling function.
S2: Call string copy function to copy the string Str1 to Dest.
S3: Go to end of the Dest string.
S4: Transfer the characters of string Str2 to Dest from the new position in string Dest.
S5: Display the concatenated string i.e, String Dest.
S6: Return to the main function .

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>

char *newstrcpy(char *d, char *s)
{
        char *p=d;
        while(*d++=*s++);  // note ;
         return p;
}


char* newstrcat(char *d, char *s)
{
         char *p=d;
         while (*d++); // note ;
                  d--;
         while(*d++=*s++); // note ;
         return p;
}

void main()
{
         char source1[25],source2[25],dest[50];
```

```
        clrscr();
        printf("\n\n\t enter a source string1\n");
        scanf("%s",source1);
        printf("\n\n\t enter a source string2\n");
        scanf("%s",source2);
        newstrcpy(dest,source1);
        printf("\n\n\tsource1 copied to dest:%s",dest);
        newstrcat(dest,source2);
        printf("\n\n\tsource2 is concatenated to dest:%s",dest);
        getch();
}
```

**OUTPUT**

enter a source string1 reva

enter a source string2 institute

source1 copied to dest: reva

source2 is concatenated to dest: revainstitute

3. Write a C Program, which accepts the Internet Protocol (IP) address in decimal dot format (ex. 153.18.8.105) and converts it into 32-bit long integer (ex. 2568095849) using strtok library function and unions.

**ALGORITHM FOR IP ADDRESS PROGRAM.**

MAIN FUNCTION()

S1: Accept IP Address from the user as a string say ipstr.
S2: Split each string token from the string ipstr where the string token is separated by dot "." as delimiter or the end of ipstr is reached.
S3: Convert each string token to an 8-bit integer number and store each number in a Contiguous memory area.
S4: Retrieve the 32-bit long integer from the stored memory area as the required result and display the same.
S5: Stop.

**PROGRAM**

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
```

```
{
        char *q,ipstr[17];
        int p,i;
        union
        {
                unsigned long int ip;
                unsigned char  a[4];
         }x;
        clrscr();
        printf("\n\n\tEnter an IP Address: ");
        scanf("%s",ipstr);
        q=strtok(ipstr,".");
        p=atoi(q);
        x.a[3]=p;
        for(i=2;i>=0;i--)
        {
                q=strtok(NULL,".");
                p=atoi(q);
                x.a[i]=p;
         }
        printf("\n\n\tLong Integer: %lu",x.ip);
        getch();
 }
```

**Output**

Enter an IP Address: 153.18.8.105

Long Integer: 2568095849

4. Write a C program to construct a stack of integers and to perform the following operations on it.

a) Push. b)Pop. c) Display.

The program should print appropriate messages for stack overflow, stack underflow and stack empty.

## ALGORITHM FOR STACK USING ARRAYS

MAIN FUNCTION()

S1: Initialize the stack using array structure.
S2: Initialize the stack pointer called top to -1.
S3: Read the operation to be performed on stack from the keyboard.
S4: If the operation specified as quit then go to step6 else go to step5.
S5: If the operation specified is push then call the push function, If the operation specified is pop then call the pop function, If the operation specified is display then call the display function,
S6: Stop.

PUSH OPERATION()

S1: Check if the stack is full, if yes display overflow or suitable message else go to step2.
S2: Read the element to be inserted from the keyboard.
S3: Increment the top by one and insert the element at top position in the stack.
S4: Return to the main function .

POP OPERATION()

S1: Check if the stack is empty, if yes stop and print appropriate message else got to step2.
S2: Display the top most element of the stack.
S3: Decrement top by one.
S4: Return to the main function .

DISPLAY OPERATION()

S1: Check if the stack is empty, if yes stop and display suitable message else go to step 2.
S2: Display the elements of the stack from the top most element to first element of the stack.
S3: Return to the main function .

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 3

int Stack[MAX];
int top;

void Pop()
{
        if(top==-1)
```

```
                printf("\n\n\tStack UnderFlow...");
        else
                printf("\n\n\n\tThe Element Popped is: %d",Stack[top--]);
}

void Push()
{
        if(top==MAX-1)
                printf("\n\n\tStack OverFlow...");
        else
        {
                printf("\n\n\tEnter an Element to Push into Stack: ");
                scanf("%d",&Stack[++top]);
         }
}

void Display()
{
        if(top==-1)
        printf("\n\n\tStack is Empty, No Elements to Display...");
        else
        {
                 int i;
                 printf("\n\nElements of the Stack are:");
                 for(i=top;i>=0;i--)
                 printf("\n\t\t\t\t%d",Stack[i]);
                 printf("\n\n\n\tTotal Number of Elements in the Stack are: %d",top+1);
 }
}




void main()
{
         int choice;
         clrscr();
         top=-1;
         while(1)
         {
                printf("\n\n\n\t1.Push...\t2.Pop...\t3.Display...\t4.Exit...");
                printf("\n\n\n\tEnter Your Choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
```

```
                        case 1: Push(); break;
                        case 2: Pop();  break;
                        case 3: Display(); break;
                        case 4: exit(0);
                        default: printf("\n\n\n\tEnter proper Choice....");
                }
        }
}
```

**OUTPUT**

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Push into Stack: 23

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Push into Stack: 78

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Push into Stack: 90

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 1

Stack OverFlow...

1.Push... 2.Pop... 3.Display... 4.Exit... Enter Your Choice: 4

5. Write a C program to convert and print a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide).

**ALGORITHM TO CONVERT INFIX EXPRESSION TO POSTFIX EXPRESSION**

MAIN FUNCTION ()

S1: Declare a data structure called stack to maintain a stack of operands .
S2: Accept Infix expression in to a string say X from user and declare a string say Y to represent postfix expression .
S3: Call convert function .
S4: Stop .

CONVERT FUNCTION()

S1: Declare i and j as two variables.
S2: Scan first symbol of infix expression and call the symbol as symb.
S3: If symb is operand put it at the end of postfix string else goto next step.
S4: If stack is not empty call precedence function with symb as parameter.
S5: If stack is not empty and precedence returns true then pop the top symbol from stack and put it at the end of postfix .
S6: Repeat step4 through step5 till stack becomes empty or precedence fails .
S7: If top symbol is '(' and scanned symbol is ')' then pop the top symbol from stack and discard the scanned symbol else put the scanned symbol on top of stack.
S8: Scan the next symbol from infix and repeat the steps from step3 through step10 till end of infix string is reached .
S9: If stack is not empty then transfer all its contents to end of postfix string .
S10: Put string terminator at the end of postfix string .
S11: Return to the main function .

PRECEDENCE FUNCTION()

S1: If top symbol in stack is $,* or / and scan symbol is *,/,+ or – return true.
S2: If top symbol is not '(' and scan symbol is ')' return true.
S3: If the top symbol is + or – and scan symbol is + or – return true.
S4: Return false.

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
```

```
#include<stdlib.h>
#include<ctype.h>

char Stack[25]; //Operator stack
int top;

char Infix[25],Postfix[25];

void Convert()
{
        int i=0,j=0;
        while(Infix[i]!=0)
        {
                char ch=Infix[i++];
                if(isalpha(ch)) Postfix[j++]=ch;
                else
                {
                        while(top!=-1&&Precedence(Stack[top],ch))
                                Postfix[j++]=Stack[top--];
                         if(top==-1||ch!=')')
                                Stack[++top]=ch;
                  }
        }
        while(top!=-1)
        {
                char c=Stack[top--];
                if(c!='(')
                Postfix[j++]=c;
        }
        Postfix[j]='\0';
}

int Precedence(char top,char cur)
{
        if(top!='('&&cur==')') return 1;
         else if((top=='$'||top=='*'||top=='/')&&(cur=='*'||cur=='/'||cur=='+'||cur=='-'))
return 1;
         else if((top=='+'||top=='-')&&(cur=='+'||cur=='-')) return 1;
         else return 0;
}
```

```
void main()
{
        clrscr(); top=-1;
        printf("\n\n\t\tEnter the Valid Infix Expression: ");
        scanf("%s",Infix);
        Convert();
        printf("\n\n\t\tThe Infix Expression   : %s",Infix);
        printf("\n\n\t\tThe Postfix Expression : %s",Postfix);
        getch();
}
```

**OUTPUT**

Enter the Valid Infix Expression: (a+b * c) / (d$e)

The Infix Expression : (a+b * c) / (d$e)

The Postfix Expression : abc*+de$/

6. Write a C program to evaluate a valid suffix/postfix expression-using stack. Assume that the suffix/postfix expression is read as a single line consisting of non-negative

single digit operands and binary arithmetic operators. The arithmetic operators are + (ADD), -(SUBTRACT), *(MULTIPLY) and /(DIVIDE).

## ALGORITHM FOR EVALUATION OF POSTFIX EXPRESSION

MAIN FUNCTION()

S1: Initialize stack called postfix using array data structure, initialize top pointer to -1.
S2: Read a valid postfix expression from key board to a string called s.
S3: Scan each and every character from the string. repeat the step step4 through step6 till end of string is reached.
S4: If the scanned symbol is an alphabet read the respective value from the keyboard and store it (push) in stack else if it is an operator go to step5
S5: Remove the top most two values from the stack and store in two variables op1 and op2 respectively.
S6: Operate the scanned operator on these two values and store the result back to stack.
S7: Check if there is more than on element in stack, if yes display invalid postfix expression and quit else go to step8
S8: Pop the last element from stack and display
S9: Stop.

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<ctype.h>
#include<math.h>

float Stack[25]; //Operand Stack...
int top;
char Postfix[25];
int flag;

float Evaluate()
{
        int i=0;
        while(Postfix[i]!='\0')
        {
                char ch=Postfix[i++];
                if(isalpha(ch))
                {
                        printf("\n\n\tEnter value of %c: ",ch);
                        scanf("%f",&Stack[++top]);
                }
                else
                {
                        if(top<1) flag=0;
                        else
                        {
                                float op1,op2;
                                op2=Stack[top--];
                                op1=Stack[top--];
                                switch(ch)
                                {
                                        case '+': Stack[++top]=op1+op2; break;
                                        case '-': Stack[++top]=op1-op2; break;
                                        case '*': Stack[++top]=op1*op2; break;
                                        case '/': Stack[++top]=op1/op2; break;
                                        case '$':
                                        case '^': Stack[++top]=pow(op1,op2); break;
                                        default: printf("\n\n\tInvalid Character: %c",ch);
flag=0;
                                }
                        }
                }
        }
```

```
        if(top!=0) flag=0;
        return Stack[top--];
}


void main()
{
        float result;
        clrscr();
        top=-1;
        flag=1;
        printf("\n\n\n\t\tEnter the Valid Postfix Expression: ");
        scanf("%s",Postfix);
        result=Evaluate();

                printf("\n\n\tInvalid Postfix Expression....");
        if(flag==1)
                printf("\n\n\tThe Result of Postfix Expression %s is: %.3f",Postfix,result);
        else getch();
}
```

**OUTPUT**

Enter the Valid Postfix Expression: ab+c/

Enter value of a: 4

Enter value of b: 2

Enter value of c: 3

The Result of Postfix Expression ab+c/ is: 2.000

7. Write a C program to simulate the working of a queue of integers using an array. Provide the following operations.
   a) Insert b) Delete c) Display

**ALGORITHM FOR QUEUE PROGRAM USING ARRAY DATA STURCTURE**

MAIN FUNCTION()

S1: Initialize the Queue using array data structure.
S2: Initialize the queue pointers called front and rear to 0 and -1 respectively.
S3: Read the operation to be performed on the queue from the keyboard.
S4: If the operation specified as quit then go to step6 else go to step5.
S5: If the operation specified is insert then call insert function, if the operation says remove call the remove function, if the operation says display then call the display function.
S6: Stop

INSERT FUNCTION()

S1: Check if the queue is full if yes display suitable message else go to step2.
S2: Read the elements to be inserted from the keyboard.
S3: Increment the rear by one and insert the element.
S4: Return to the main program.

REMOVE OPERATION()

S1: Check if the queue is empty if yes display suitable message else go to step2.
S2: Remove and display the removed element.
S3: Increment the front by one.
S4: Return to the main program..

DISPLAY OPERATION()

S1 : Check if the queue is empty if yes display suitable message else go to step2
S2: Display the elements of the queue from the front to the rear position of the queue.
S3: Return to the main program.

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 5

int Queue[MAX];
int front,rear;

void Remove()
{
        if(rear<front)
                printf("\n\n\tQueue UnderFlow...");
        else
                printf("\n\n\n\tThe Element Removed from Queue is:
%d",Queue[front++]);
}

void Insert()
{
        if(rear==MAX-1)
                printf("\n\n\tQueue OverFlow...");
        else
        {
                 printf("\n\n\tEnter an Element to Insert into Queue: ");
                 scanf("%d",&Queue[++rear]);
         }
}

void Display()
{
        if(rear<front)
                printf("\n\n\tQueue is Empty, No Elements to Display...");
        else
```

```
        {
                int i;
                printf("\n\nElements of the QUEUE are:");
                for(i=front;i<=rear;i++)
                        printf("\n\t\t\t\t\t%d",Queue[i]);
                printf("\n\n\n\tTotal Number of Elements in Queue are: %d",rear-
front+1);
         }
}



void main()
{
        int choice;
        clrscr();
        rear=-1;
        front=0;
        while(1)
        {
                printf("\n\n\n\t1.Insert...\t2.Remove...\t3.Display...\t4.Exit...");
                printf("\n\n\n\tEnter Your Choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: Insert(); break;
                        case 2: Remove();  break;
                        case 3: Display(); break;
                        case 4: exit(0);
                        default: printf("\n\n\n\tEnter proper Choice....");
                }
        }
}
```

**OUTPUT**

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Queue: 12

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Queue: 213

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Queue: 45

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Queue OverFlow...
1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 12
213
45
Total Number of Elements in Queue are: 3

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Queue is: 12

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Queue is: 213

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Queue is: 45

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

Queue UnderFlow...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Queue is: 12

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Queue is: 23

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Queue is: 45

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

Queue UnderFlow...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 4

8. Write a C program to simulate the working of a circular queue of integers using an array. Provide the following operations.
   a) Insert b)Delete c) Display

**ALGORITHM FOR CIRCULAR QUEUE**

MAIN FUNCTION()

S1: Initialize a queue called CQ using array data structure.
S2: Consider two pointers front, rear and initialize, read the max size of queue from user.
S3: Consider two pointers front and rear to make the deleting and inserting end respectively and initialize both the pointer to max size of CQ.
S4: Read the operation to be performed on CQ from the keyboard.
S5: If the operation specified is insert then call the insert function, else If the operation specified is remove then call the remove function, If the operation specified is display then call the display function, If the operation specified is quit then go to step6.
S6: Stop.

INSERT FUNCTION()

S1: Check if the CQ is full, if yes then display CQ overflow condition and quit else go to step 2.
S2: Check if rear is equal to max size, if yes then reassign rear pointer to zero else increment rear pointer.
S3: Read the element to be inserted from keyboard and store at the rear position of the CQ.
S4: Return to the main program

REMOVE FUNCTION()

S1: Check if CQ is empty if yes then display underflow condition, if yes then display underflow condition and quit else go to step 2.

S2: Check if front is reached the max size of CQ if yes then reassign the front to zero else increment the front pointer.
S3: Remove the element at front position of CQ and display.
S4: Return to the main program

DISPLAY FUNCTION()

S1: Check if CQ is empty if yes then display underflow condition and goto step3 else goto step2.
S2: Display all the elements of the CQ from front to the rear position.
S3: Return to the main program

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 5

int CQueue[MAX];
int front,rear;



void Remove()
{
        if(rear==front)
                printf("\n\n\tCircular Queue UnderFlow...");
        else
        {
                if(front==MAX-1)
                        front=0;
                else
                        front++;
                printf("\n\n\n\tThe Element Removed from Circular Queue is:
%d",CQueue[front]);
```

```
        }
}


void Insert()
{
        if((MAX+rear-front)%MAX==MAX-1)
                printf("\n\n\tCircular Queue OverFlow...");
        else
        {
                if(rear==MAX-1)
                        rear=0;
                 else
                        rear++;
                printf("\n\n\tEnter an Element to Insert into Circular Queue: ");
                scanf("%d",&CQueue[rear]);
         }
}









void Display()
{
        if(rear==front)
                printf("\n\n\tCircular Queue is Empty, No Elements to Display...");
        else
        {
                int i,j;
                if(front==MAX-1)
                        i=0;
                else
                        i=front+1;
                printf("\n\nElements of the Circular Queue are:");
                for(j=1;j<=(MAX+rear-front)%MAX;j++)
                {
                        printf("\n\t\t\t\t\t%d",CQueue[i++]);
                        if(i==MAX)
                                i=0;
                }
```

```
                printf("\n\n\n\tTotal Number of Elements in the Circular Queue are:
%d",j-1);
        }
}


void main()
{
        int choice;
        clrscr();
        rear=front=MAX-1;
        while(1)
        {
                printf("\n\n\n\t1.Insert...\t2.Remove...\t3.Display...\t4.Exit...");
                printf("\n\n\n\tEnter Your Choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: Insert(); break;
                        case 2: Remove();  break;
                        case 3: Display(); break;
                        case 4: exit(0);
                        default: printf("\n\n\n\tEnter proper Choice....");
                }
 }
}
```

**OUTPUT**

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Circular Queue: 11

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Circular Queue: 22

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Circular Queue: 33

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Circular Queue: 44

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Circular Queue OverFlow...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 3

Elements of the Circular Queue are:

11
22
33
44

Total Number of Elements in the Circular Queue are: 4

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Circular Queue is: 11

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Circular Queue is: 22

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Circular Queue is: 33

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Circular Queue is: 22

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Circular Queue is: 33

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Circular Queue is: 44

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

Circular Queue UnderFlow...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 3

Circular Queue is Empty, No Elements to Display...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 4

9. Write a C Program using dynamic variables and pointers, to construct a singly linked list consisting of the following information n each node: student id (integer), student name (character string) and semester (integer). The operations to be supported are:
   a. The insertion operation
      i. At the front of a list
      ii. At the back of the list
      iii. At any position in the list
   b. Deleting a node based on student id. If the specified node is not present in the list an error message should be displayed. Both the options should be demonstrated.
   c. Searching a node based on student id and update the information content. If the specified node is not present in the list an error message should be displayed. Both situations should be displayed.
   d. Displaying all the nodes in the list.(Note: Only one set of operations among a, b and c with d may be asked in the examination).

**ALGORITHM FOR STUDENT INFORMATION PROGRAM**

MAIN FUNCTION()

S1 : Initialize a list using linked list data structure where head is called as te start node of the list .
S2 : Read the operation fro the keyboard and move to the respective function .
S3 : Stop .

INSERT FRONT FUNCTION()

S1 : Call the getnode function to create the node dynamically .
S2 : Assign the head node address to the link field of newnode .
S3 : Assign the newnode address to the headnode .
S4: Return to the main program

INSERT END FUNCTION().

S1 : Call getnode function to create a newnode dynamically.
S2: Check if the list empty, If yes then the newnode is the head node, assign the newnode address to the head node else goto step step3.
S3: Search for the presence of the last node in the list and then assign the newnode address to the link field of the last node.
S4: Return to the main program

INSERT AT A GIVEN POSITION()

S1: Read the value of the position from the keyboard.
S2: Check if the position value is a positive integer greater than zero and less than Max size of the list, if no display invalid position and quit else goto step3.
S3: If the position is 1 then call insert front function else goto step step4.
S4: If the position is equal to MAX size then call insert end function else goto step5.
S5: Call getnode function to create a node dynamically.
S6: Scan till the specified position is reached in the list and assign the respective node address to the temporary node called P.
S7: Assign the link field of P to the link field of newnode.
S8: Assign the newnode address to the link field of P.
S9: Return to the main program

DELETE FUNCTION()
S1: Check if the list id empty, if yes then display list empty and quit else goto step2.
S2: Read the element to be deleted form the list.
S3: Scan the occurrence of the element in the list if the element not found then display element not found and quit else goto step4.
S4: Assign the node at which the element is found to variable called p.
S5: Display the contents of the node P.
S6: Assign P's next node address to the link field of P's previous node.
S7: Decrement count by one.
S8: Free (P) or Deallocate P's memory.
S9: Return to the main program.

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 5

struct Node
{
        int id;
        char name[25];
        int sem;
        struct Node *next;
};
typedef struct Node NODE;

NODE *head;
int count;

NODE *Getnode()
{
        NODE *newnode;
        newnode=(NODE *)malloc(sizeof(NODE));
        newnode->next=NULL;
        printf("\n\tEnter Student Id: "); scanf("%d",&newnode->id);
        printf("\n\tEnter Name      : "); scanf("%s",newnode->name);
        printf("\n\tEnter Semester  : "); scanf("%d",&newnode->sem);
        count++;
        return newnode;
}
```

**……**

1. Write a C program to create a sequential file with at least five records, each records, each record having the structure shown below :

| USN | Name | Marks 1 | Marks 2 | Marks 3 |
|---|---|---|---|---|
| Non-zero Positive integer | 25 Characters | Positive integer | Positive integer | Positive integer |

Write necessary functions

a) To display all the records in the file.

b) To search for a specific record based on the USN. In case the required record is not found, suitable message should be displayed. Both the options in this case must be demonstrated.

**ALGORITHM FOR FILES PROGRAM.**

MAIN FUNCTION()

S1: Read the filename from user .
S2: Read the operations to be performed from the keyboard .
S3: If the operation specified is insert go to the insert function , if the operation specified is search go to the search function, if the operation specified is display go to the display function, if the operation specified is exit go to step 4.
S4: Stop.

INSERT FUNCTION()

S1: Open the file in the write mode ,if the file specified is not found or unable to open then display error message and go to step5 , else go to step2.
S2: Read the no: of records N to be inserted to the file .
S3: Repeat the step4 N number of times .
S4: Read the details of each student from the keyboard and write the same to the file .
S5: Close the file .
S6: Return to the main function .

SEARCH FUNCTION()

S1: Open the file in the read mode ,if the file specified is not found or unable to open then display error message and go to step6 , else go to step2.
S2: Read the USN number of the student whose details need to be searched .
S3: Read each student record from the file.
S4: Compare the students USN number scanned from file with USN number specified by the user.
S5: If they are equal then display the details of the searched record else display required USN number not found message and go to step6.
S6: Close the file.
S7: Return to the main function.

DISPLAY FUNCTION()

S1: Open the specified file in read mode.
S2: If the file is unable to open or not found then display error message and go to step4 else go to step3.
S3: Scan all the student details one by one from file and display the same at the console until end of file is reached.
S4: Close the file.
S5: Return to the main function .

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>

struct Student
{
        char name[25];
         int  usn;
         int m1,m2,m3;
};

char filename[15];
int n;
struct Student s;

void Insert()
{
        FILE *fp;
        int i;
        printf("\n\n\tEnter how many Records: ");
        scanf("%d",&n);
        fp=fopen(filename,"w");
        if(fp==NULL) printf("\n\n\tError in Creating the File....");
        else
                for(i=0;i<n;i++)
                {
                        printf("\nEnter Details of Student-%d: ",i+1);
                        printf("\n\t\t\tName: "); scanf("%s",s.name); fprintf(fp,"%s
                ",s.name);
                        printf("\n\t\t\tUsn : "); scanf("%d",&s.usn);  fprintf(fp,"%d
                ",s.usn);
                        printf("\n\t\t\tM1  : "); scanf("%d",&s.m1); fprintf(fp,"%d ",s.m1);
                        printf("\n\t\t\tM2  : "); scanf("%d",&s.m2); fprintf(fp,"%d ",s.m2);
                        printf("\n\t\t\tM3  : "); scanf("%d",&s.m3); fprintf(fp,"%d ",s.m3);
                }
         fclose(fp);
```

```
}
void Search()
{
        if(n==0)
                printf("\n\n\tNo Records to Search...");
        else
        {
                 FILE *fp;
                 int i,flag=0,usn;
                fp=fopen(filename,"r");
                if(fp==NULL)
                        printf("\n\n\tERROR!! in Opening the File....");
                else
                {
                        printf("\n\n\tEnter the Usn to be Searched: ");
                        scanf("%d",&usn);
                        for(i=0;i<n;i++)
                        {
                                fscanf(fp,"%s",s.name);
                                fscanf(fp,"%d",&s.usn);
                                fscanf(fp,"%d",&s.m1);
                                fscanf(fp,"%d",&s.m2);
                                fscanf(fp,"%d",&s.m3);
                                if(usn==s.usn)
                                {
                                        flag=1;
                                         break;
                                }
                        }
                        if(flag==1)
                        {
                                 printf("\nRecord Exist And Details are:");
                                 printf("\n\t\t\t\tName: %s",s.name);
                                 printf("\n\t\t\t\tUsn : %d",s.usn);
                                 printf("\n\t\t\t\tM1  : %d",s.m1);
                                 printf("\n\t\t\t\tM2  : %d",s.m2);
                                 printf("\n\t\t\t\tM3  : %d",s.m3);
                        }
                        else
                                printf("\n\n\tRecord having Usn %d Does Not
Exist...\n\n",usn);
                }
                fclose(fp);
        }
}
```

```
void Display()
{
        if(n==0)
                printf("\n\n\tNo Records to Display....\n\n");
        else
        {
                int i;
                FILE *fp=fopen(filename,"r");
                if(fp==NULL)
                        printf("\n\n\tERROR!! in Opening File...");
                else
                        for(i=0;i<n;i++)
                        {
                                printf("\nDetails of Student-%d: ",i+1);
                                fscanf(fp,"%s",s.name); printf("\n\t\t\tName: %s",s.name);
                                fscanf(fp,"%d",&s.usn);  printf("\n\t\t\tUsn : %d",s.usn);
                                fscanf(fp,"%d",&s.m1); printf("\n\t\t\tM1  : %d",s.m1);
                                fscanf(fp,"%d",&s.m2); printf("\n\t\t\tM2  : %d",s.m2);
                                fscanf(fp,"%d",&s.m3); printf("\n\t\t\tM3  : %d",s.m3);
                        }
                fclose(fp);
        }
}


void main()
{
        int ch;
        clrscr();
        printf("\n\n\tEnter the File Name: ");
        scanf("%s",filename);
        while(1)
        {
                printf("\n\n\t1.Insert...\t2.Search...\t3.display...\t4.Exit...");
                printf("\n\n\tEnter Your Choice: ");
                scanf("%d",&ch);
                 switch(ch)
                 {
                        case 1: Insert();  break;
                        case 2: Search(); break;
                        case 3: Display(); break;
                        case 4: exit(0);
                        default: printf("\n\n\tEnter Proper Choice...");
                }
```

```
        }
}
```

**OUTPUT**

```
Enter the File Name: test

1.Insert...  2.Search...  3.display...  4.Exit...

Enter Your Choice: 1

Enter how many Records: 1

Enter Details of Student-1:
        Name: nikhil

        Usn : 100

        M1  : 23

        M2  : 24

        M3  : 25


1.Insert...  2.Search...  3.display...  4.Exit...

Enter Your Choice: 2


   Enter the Usn to be Searched: 100

Record Exist And Details are:
        Name: nikhil
        Usn : 100
        M1  : 23
        M2  : 24
        M3  : 25

1.Insert...  2.Search...  3.display...  4.Exit...

   Enter Your Choice: 2

   Enter the Usn to be Searched: 101

   Record having Usn 101 Does Not Exist..
```

2. Write and demonstrate the following C functions:
   a. newStrCpy that does the same job as strcpy
   b. newStrCat that does the same job as strcat without using any library functions.

**ALGORITHM FOR STRING COPY AND STRING CONCATINATION**

MAIN FUNCTION()

S1: Read string (source string) called Str1 from the keyboard.
S2: Call the string copy function with Str1 as the input parameter.
S3: Read another string called Str2 from keyboard.
S4: Call string concatenate function with Str1 and Str2 as input parameters.
S5: Stop

STRING COPY FUNCTION()

S1: Consider an empty string called Dest and accept Str1 string from calling function.
S2: Consider two variables called i and j and initialize i to the first character of source string and j to the first free location of the string d.
S3: Repeat steps step4 through step5 until end of Str1 string is reached.
S4: Copy the value of i to the jth location of the destination string Dest.
S5: Increment value and assign next character of string Str1 to i.
S6: Display string Dest. [copied string].
S7: Return to the main function .

STRING CONCATENATE FUNCTION()

S1: Consider an empty string called Dest and accept Str1 and Str2 strings from the calling function.
S2: Call string copy function to copy the string Str1 to Dest.
S3: Go to end of the Dest string.
S4: Transfer the characters of string Str2 to Dest from the new position in string Dest.
S5: Display the concatenated string i.e, String Dest.
S6: Return to the main function .

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>

char *newstrcpy(char *d, char *s)
{
        char *p=d;
        while(*d++=*s++);  // note ;
         return p;
}


char* newstrcat(char *d, char *s)
{
         char *p=d;
         while (*d++); // note ;
                  d--;
         while(*d++=*s++); // note ;
         return p;
}

void main()
{
         char source1[25],source2[25],dest[50];
         clrscr();
         printf("\n\n\t enter a source string1\n");
         scanf("%s",source1);
         printf("\n\n\t enter a source string2\n");
         scanf("%s",source2);
         newstrcpy(dest,source1);
         printf("\n\n\tsource1 copied to dest:%s",dest);
         newstrcat(dest,source2);
         printf("\n\n\tsource2 is concatenated to dest:%s",dest);
         getch();
}
```

**OUTPUT**

```
         enter a source string1  reva
```

enter a source string2 institute

source1 copied to dest: reva

source2 is concatenated to dest: revainstitute

3. Write a C Program, which accepts the Internet Protocol (IP) address in decimal dot format
(ex. 153.18.8.105) and converts it into 32-bit long integer (ex. 2568095849) using strtok
library function and unions.

**ALGORITHM FOR IP ADDRESS PROGRAM.**

MAIN FUNCTION()

S1: Accept IP Address from the user as a string say ipstr.
S2: Split each string token from the string ipstr where the string token is separated by dot "." as delimiter or the end of ipstr is reached.
S3: Convert each string token to an 8-bit integer number and store each number in a Contiguous memory area.
S4: Retrieve the 32-bit long integer from the stored memory area as the required result and display the same.
S5: Stop.

**PROGRAM**

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
        char *q,ipstr[17];
        int p,i;
        union
        {
                unsigned long int ip;
                unsigned char  a[4];
         }x;
        clrscr();
        printf("\n\n\tEnter an IP Address: ");
        scanf("%s",ipstr);
        q=strtok(ipstr,".");
        p=atoi(q);
        x.a[3]=p;
        for(i=2;i>=0;i--)
        {
```

```
            q=strtok(NULL,".");
            p=atoi(q);
            x.a[i]=p;
        }
        printf("\n\n\tLong Integer: %lu",x.ip);
        getch();
}
```

**Output**

Enter an IP Address: 153.18.8.105

Long Integer: 2568095849 4. Write a C program to construct a stack of integers and to perform the following operations on it.

a) Push. b)Pop. c) Display.

The program should print appropriate messages for stack overflow, stack underflow and stack empty.

## ALGORITHM FOR STACK USING ARRAYS

MAIN FUNCTION()

S1: Initialize the stack using array structure.
S2: Initialize the stack pointer called top to -1.
S3: Read the operation to be performed on stack from the keyboard.
S4: If the operation specified as quit then go to step6 else go to step5.
S5: If the operation specified is push then call the push function, If the operation specified is pop then call the pop function, If the operation specified is display then call the display function,
S6: Stop.

PUSH OPERATION()

S1: Check if the stack is full, if yes display overflow or suitable message else go to step2.
S2: Read the element to be inserted from the keyboard.
S3: Increment the top by one and insert the element at top position in the stack.
S4: Return to the main function .

POP OPERATION()

S1: Check if the stack is empty, if yes stop and print appropriate message else got to step2.
S2: Display the top most element of the stack.
S3: Decrement top by one.
S4: Return to the main function .

DISPLAY OPERATION()

S1: Check if the stack is empty, if yes stop and display suitable message else go to step 2.

S2: Display the elements of the stack from the top most element to first element of the stack.
S3: Return to the main function .

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 3

int Stack[MAX];
int top;

void Pop()
{
        if(top==-1)
                printf("\n\n\tStack UnderFlow...");
        else
                printf("\n\n\n\tThe Element Popped is: %d",Stack[top--]);
}

void Push()
{
        if(top==MAX-1)
                printf("\n\n\tStack OverFlow...");
        else
        {
                printf("\n\n\tEnter an Element to Push into Stack: ");
                scanf("%d",&Stack[++top]);
         }
}

void Display()
{
```

```
        if(top==-1)
        printf("\n\n\tStack is Empty, No Elements to Display...");
        else
        {
                int i;
                printf("\n\nElements of the Stack are:");
                for(i=top;i>=0;i--)
                printf("\n\t\t\t\t%d",Stack[i]);
                printf("\n\n\n\tTotal Number of Elements in the Stack are: %d",top+1);
 }
}



void main()
{
        int choice;
        clrscr();
        top=-1;
        while(1)
        {
                printf("\n\n\n\t1.Push...\t2.Pop...\t3.Display...\t4.Exit...");
                printf("\n\n\n\tEnter Your Choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: Push(); break;
                        case 2: Pop();  break;
                        case 3: Display(); break;
                        case 4: exit(0);
                        default: printf("\n\n\n\tEnter proper Choice....");
                }
        }
}
```

**OUTPUT**

```
        1.Push...       2.Pop...        3.Display...    4.Exit...

        Enter Your Choice: 1

        Enter an Element to Push into Stack: 23

        1.Push...       2.Pop...        3.Display...    4.Exit...
```

Enter Your Choice: 1

Enter an Element to Push into Stack: 78

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Push into Stack: 90

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 1

Stack OverFlow...

1.Push... 2.Pop... 3.Display... 4.Exit... Enter Your Choice: 4

5. Write a C program to convert and print a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide).

**ALGORITHM TO CONVERT INFIX EXPRESSION TO POSTFIX EXPRESSION**

MAIN FUNCTION ()

S1: Declare a data structure called stack to maintain a stack of operands .
S2: Accept Infix expression in to a string say X from user and declare a string say Y to represent postfix expression .
S3: Call convert function .
S4: Stop .

CONVERT FUNCTION()

S1: Declare i and j as two variables.
S2: Scan first symbol of infix expression and call the symbol as symb.
S3: If symb is operand put it at the end of postfix string else goto next step.
S4: If stack is not empty call precedence function with symb as parameter.
S5: If stack is not empty and precedence returns true then pop the top symbol from stack and put it at the end of postfix .
S6: Repeat step4 through step5 till stack becomes empty or precedence fails .
S7: If top symbol is '(' and scanned symbol is ')' then pop the top symbol from stack and discard the scanned symbol else put the scanned symbol on top of stack.
S8: Scan the next symbol from infix and repeat the steps from step3 through step10 till end of infix string is reached .
S9: If stack is not empty then transfer all its contents to end of postfix string .

S10: Put string terminator at the end of postfix string .
S11: Return to the main function .

PRECEDENCE FUNCTION()

S1: If top symbol in stack is $,* or / and scan symbol is *,/,+ or – return true.
S2: If top symbol is not '(' and scan symbol is ')' return true.
S3: If the top symbol is + or – and scan symbol is + or – return true.
S4: Return false.

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<ctype.h>

char Stack[25]; //Operator stack
int top;

char Infix[25],Postfix[25];

void Convert()
{
        int i=0,j=0;
        while(Infix[i]!=0)
        {
                char ch=Infix[i++];
                if(isalpha(ch)) Postfix[j++]=ch;
                else
                {
                        while(top!=-1&&Precedence(Stack[top],ch))
                                Postfix[j++]=Stack[top--];
                         if(top==-1||ch!=')')
                                Stack[++top]=ch;
```

```
                }
        }
        while(top!=-1)
        {
                char c=Stack[top--];
                if(c!='(')
                Postfix[j++]=c;
        }
        Postfix[j]='\0';
}

int Precedence(char top,char cur)
{
        if(top!='('&&cur==')') return 1;
         else if((top=='$'||top=='*'||top=='/')&&(cur=='*'||cur=='/'||cur=='+'||cur=='-'))
return 1;
         else if((top=='+'||top=='-')&&(cur=='+'||cur=='-')) return 1;
         else return 0;
}




void main()
{
         clrscr(); top=-1;
         printf("\n\n\t\tEnter the Valid Infix Expression: ");
         scanf("%s",Infix);
         Convert();
         printf("\n\n\t\tThe Infix Expression   : %s",Infix);
         printf("\n\n\t\tThe Postfix Expression : %s",Postfix);
         getch();
}
```

**OUTPUT**

```
Enter the Valid Infix Expression: (a+b * c) / (d$e)
```

The Infix Expression : (a+b * c) / (d$e)

The Postfix Expression : abc*+de$/

6. Write a C program to evaluate a valid suffix/postfix expression-using stack. Assume that the suffix/postfix expression is read as a single line consisting of non-negative single digit operands and binary arithmetic operators. The arithmetic operators are + (ADD), -(SUBTRACT), *(MULTIPLY) and /(DIVIDE).

**ALGORITHM FOR EVALUATION OF POSTFIX EXPRESSION**

MAIN FUNCTION()

S1: Initialize stack called postfix using array data structure, initialize top pointer to -1.
S2: Read a valid postfix expression from key board to a string called s.
S3: Scan each and every character from the string. repeat the step step4 through step6 till end of string is reached.
S4: If the scanned symbol is an alphabet read the respective value from the keyboard and store it (push) in stack else if it is an operator go to step5
S5: Remove the top most two values from the stack and store in two variables op1 and op2 respectively.
S6: Operate the scanned operator on these two values and store the result back to stack.
S7: Check if there is more than on element in stack, if yes display invalid postfix expression and quit else go to step8
S8: Pop the last element from stack and display
S9: Stop.

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<ctype.h>
#include<math.h>

float Stack[25]; //Operand Stack...
int top;
char Postfix[25];
int flag;

float Evaluate()
{
        int i=0;
        while(Postfix[i]!='\0')
        {
                char ch=Postfix[i++];
                if(isalpha(ch))
                {
                        printf("\n\n\tEnter value of %c: ",ch);
                        scanf("%f",&Stack[++top]);
                 }
```

```
                else
                {
                        if(top<1) flag=0;
                        else
                        {
                                float op1,op2;
                                op2=Stack[top--];
                                op1=Stack[top--];
                                switch(ch)
                                {
                                        case '+': Stack[++top]=op1+op2; break;
                                        case '-': Stack[++top]=op1-op2; break;
                                        case '*': Stack[++top]=op1*op2; break;
                                        case '/': Stack[++top]=op1/op2; break;
                                        case '$':
                                        case '^': Stack[++top]=pow(op1,op2); break;
                                        default: printf("\n\n\tInvalid Character: %c",ch);
flag=0;
                                }
                        }
                }
        }
        if(top!=0) flag=0;
        return Stack[top--];
}

void main()
{
        float result;
        clrscr();
        top=-1;
        flag=1;
        printf("\n\n\n\t\tEnter the Valid Postfix Expression: ");
        scanf("%s",Postfix);
        result=Evaluate();

                printf("\n\n\tInvalid Postfix Expression....");
        if(flag==1)
                printf("\n\n\tThe Result of Postfix Expression %s is: %.3f",Postfix,result);
        else getch();
}
```

**OUTPUT**

```
 Enter the Valid Postfix Expression:  ab+c/

      Enter value of a: 4
```

Enter value of b: 2

Enter value of c: 3

The Result of Postfix Expression ab+c/ is: 2.000

7. Write a C program to simulate the working of a queue of integers using an array. Provide the following operations.
   a) Insert b) Delete c) Display

**ALGORITHM FOR QUEUE PROGRAM USING ARRAY DATA STURCTURE**

MAIN FUNCTION()

S1: Initialize the Queue using array data structure.
S2: Initialize the queue pointers called front and rear to 0 and -1 respectively.
S3: Read the operation to be performed on the queue from the keyboard.
S4: If the operation specified as quit then go to step6 else go to step5.
S5: If the operation specified is insert then call insert function, if the operation says remove call the remove function, if the operation says display then call the display function.
S6: Stop

INSERT FUNCTION()

S1: Check if the queue is full if yes display suitable message else go to step2.
S2: Read the elements to be inserted from the keyboard.
S3: Increment the rear by one and insert the element.
S4: Return to the main program.

REMOVE OPERATION()

S1: Check if the queue is empty if yes display suitable message else go to step2.
S2: Remove and display the removed element.
S3: Increment the front by one.
S4: Return to the main program..

DISPLAY OPERATION()

S1 : Check if the queue is empty if yes display suitable message else go to step2
S2: Display the elements of the queue from the front to the rear position of the queue.
S3: Return to the main program.

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 5

int Queue[MAX];
int front,rear;

void Remove()
{
        if(rear<front)
                printf("\n\n\tQueue UnderFlow...");
        else
```

```
                printf("\n\n\n\tThe Element Removed from Queue is:
%d",Queue[front++]);
}

void Insert()
{
        if(rear==MAX-1)
                printf("\n\n\tQueue OverFlow...");
        else
        {
                 printf("\n\n\tEnter an Element to Insert into Queue: ");
                 scanf("%d",&Queue[++rear]);
         }
}

void Display()
{
        if(rear<front)
                printf("\n\n\tQueue is Empty, No Elements to Display...");
        else
        {
                 int i;
                 printf("\n\nElements of the QUEUE are:");
                 for(i=front;i<=rear;i++)
                          printf("\n\t\t\t\t\t%d",Queue[i]);
                 printf("\n\n\n\tTotal Number of Elements in Queue are: %d",rear-
front+1);
         }
}


void main()
{
        int choice;
        clrscr();
        rear=-1;
        front=0;
        while(1)
        {
                 printf("\n\n\n\t1.Insert...\t2.Remove...\t3.Display...\t4.Exit...");
                 printf("\n\n\n\tEnter Your Choice: ");
                 scanf("%d",&choice);
                 switch(choice)
                 {
                          case 1: Insert(); break;
                          case 2: Remove();  break;
```

```
                        case 3: Display(); break;
                        case 4: exit(0);
                        default: printf("\n\n\n\tEnter proper Choice....");
                }
        }
}
```

**OUTPUT**

```
1.Insert...   2.Remove...   3.Display...   4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Queue: 12

1.Insert...   2.Remove...   3.Display...   4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Queue: 213


1.Insert...   2.Remove...   3.Display...   4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Queue: 45

1.Insert...   2.Remove...   3.Display...   4.Exit...

Enter Your Choice: 1
Queue OverFlow...
1.Insert...   2.Remove...   3.Display...   4.Exit...

Enter Your Choice: 12
                        213
                        45
Total Number of Elements in Queue are: 3


1.Insert...   2.Remove...   3.Display...   4.Exit...

Enter Your Choice: 2

The Element Removed from Queue is: 12
```

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Queue is: 213

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Queue is: 45

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

Queue UnderFlow...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Queue is: 12

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Queue is: 23

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Queue is: 45

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

Queue UnderFlow...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 4

8. Write a C program to simulate the working of a circular queue of integers using an array. Provide the following operations.
   a) Insert b)Delete c) Display

**ALGORITHM FOR CIRCULAR QUEUE**

MAIN FUNCTION()

S1: Initialize a queue called CQ using array data structure.
S2: Consider two pointers front, rear and initialize, read the max size of queue from user.
S3: Consider two pointers front and rear to make the deleting and inserting end respectively and initialize both the pointer to max size of CQ.
S4: Read the operation to be performed on CQ from the keyboard.

S5: If the operation specified is insert then call the insert function, else If the operation specified is remove then call the remove function, If the operation specified is display then call the display function, If the operation specified is quit then go to step6.
S6: Stop.

INSERT FUNCTION()

S1: Check if the CQ is full, if yes then display CQ overflow condition and quit else go to step 2.
S2: Check if rear is equal to max size, if yes then reassign rear pointer to zero else increment rear pointer.
S3: Read the element to be inserted from keyboard and store at the rear position of the CQ.
S4: Return to the main program

REMOVE FUNCTION()

S1: Check if CQ is empty if yes then display underflow condition, if yes then display underflow condition and quit else go to step 2.
S2: Check if front is reached the max size of CQ if yes then reassign the front to zero else increment the front pointer.
S3: Remove the element at front position of CQ and display.
S4: Return to the main program

DISPLAY FUNCTION()

S1: Check if CQ is empty if yes then display underflow condition and goto step3 else goto step2.
S2: Display all the elements of the CQ from front to the rear position.
S3: Return to the main program

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 5

int CQueue[MAX];
int front,rear;
```

```
void Remove()
{
        if(rear==front)
                printf("\n\n\tCircular Queue UnderFlow...");
        else
        {
                if(front==MAX-1)
                        front=0;
                else
                        front++;
                printf("\n\n\n\tThe Element Removed from Circular Queue is:
%d",CQueue[front]);
        }
}



void Insert()
{
        if((MAX+rear-front)%MAX==MAX-1)
                printf("\n\n\tCircular Queue OverFlow...");
        else
        {
                if(rear==MAX-1)
                        rear=0;
                 else
                        rear++;
                printf("\n\n\tEnter an Element to Insert into Circular Queue: ");
                scanf("%d",&CQueue[rear]);
         }
}







void Display()
{
        if(rear==front)
                printf("\n\n\tCircular Queue is Empty, No Elements to Display...");
        else
        {
```

```
                int i,j;
                if(front==MAX-1)
                        i=0;
                else
                        i=front+1;
                printf("\n\nElements of the Circular Queue are:");
                for(j=1;j<=(MAX+rear-front)%MAX;j++)
                {
                        printf("\n\t\t\t\t\t%d",CQueue[i++]);
                        if(i==MAX)
                                i=0;
                }
                printf("\n\n\n\tTotal Number of Elements in the Circular Queue are:
%d",j-1);
        }
}


void main()
{
        int choice;
        clrscr();
        rear=front=MAX-1;
        while(1)
        {
                printf("\n\n\n\t1.Insert...\t2.Remove...\t3.Display...\t4.Exit...");
                printf("\n\n\n\tEnter Your Choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: Insert(); break;
                        case 2: Remove();  break;
                        case 3: Display(); break;
                        case 4: exit(0);
                        default: printf("\n\n\n\tEnter proper Choice....");
                }
 }
}
```

**OUTPUT**

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Circular Queue: 11

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Circular Queue: 22

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Circular Queue: 33

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert into Circular Queue: 44

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Circular Queue OverFlow...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 3

Elements of the Circular Queue are:

11
22
33
44

Total Number of Elements in the Circular Queue are: 4

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Circular Queue is: 11

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Circular Queue is: 22

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Circular Queue is: 33

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Circular Queue is: 22

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Circular Queue is: 33

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

The Element Removed from Circular Queue is: 44

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

Circular Queue UnderFlow...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 3

Circular Queue is Empty, No Elements to Display...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 4 9. Write a C Program using dynamic variables and pointers, to construct a singly linked list consisting of the following information n each node: student id (integer), student name (character string) and semester (integer). The operations to be supported are:

a. The insertion operation

i. At the front of a list

ii. At the back of the list
iii. At any position in the list

b. Deleting a node based on student id. If the specified node is not present in the list an error message should be displayed. Both the options should be demonstrated.

c. Searching a node based on student id and update the information content. If the specified node is not present in the list an error message should be displayed. Both situations should be displayed.

d. Displaying all the nodes in the list.(Note: Only one set of operations among a, b and c with d may be asked in the examination).

**ALGORITHM FOR STUDENT INFORMATION PROGRAM**

MAIN FUNCTION()

S1 : Initialize a list using linked list data structure where head is called as te start node of the list .
S2 : Read the operation fro the keyboard and move to the respective function .
S3 : Stop .

INSERT FRONT FUNCTION()

S1 : Call the getnode function to create the node dynamically .
S2 : Assign the head node address to the link field of newnode .
S3 : Assign the newnode address to the headnode .
S4: Return to the main program

INSERT END FUNCTION().

S1 : Call getnode function to create a newnode dynamically.
S2: Check if the list empty, If yes then the newnode is the head node, assign the newnode address to the head node else goto step step3.
S3: Search for the presence of the last node in the list and then assign the newnode address to the link field of the last node.
S4: Return to the main program

INSERT AT A GIVEN POSITION()

S1: Read the value of the position from the keyboard.
S2: Check if the position value is a positive integer greater than zero and less than Max size of the list, if no display invalid position and quit else goto step3.
S3: If the position is 1 then call insert front function else goto step step4.
S4: If the position is equal to MAX size then call insert end function else goto step5.
S5: Call getnode function to create a node dynamically.
S6: Scan till the specified position is reached in the list and assign the respective node address to the temporary node called P.
S7: Assign the link field of P to the link field of newnode.
S8: Assign the newnode address to the link field of P.
S9: Return to the main program

DELETE FUNCTION()

S1: Check if the list id empty, if yes then display list empty and quit else goto step2.
S2: Read the element to be deleted form the list.

S3: Scan the occurrence of the element in the list if the element not found then display element not found and quit else goto step4.
S4: Assign the node at which the element is found to variable called p.
S5: Display the contents of the node P.
S6: Assign P's next node address to the link field of P's previous node.
S7: Decrement count by one.
S8: Free (P) or Deallocate P's memory.
S9: Return to the main program.

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 5

struct Node
{
        int id;
        char name[25];
        int sem;
        struct Node *next;
};
typedef struct Node NODE;

NODE *head;
int count;

NODE *Getnode()
{
        NODE *newnode;
        newnode=(NODE *)malloc(sizeof(NODE));
        newnode->next=NULL;
        printf("\n\tEnter Student Id: "); scanf("%d",&newnode->id);
        printf("\n\tEnter Name      : "); scanf("%s",newnode->name);
        printf("\n\tEnter Semester  : "); scanf("%d",&newnode->sem);
        count++;
        return newnode;
}



// ****************** Sub-Program A **********************

void InsertFront()
```

```
{
        NODE *newnode;
        newnode=Getnode();
        newnode->next=head;
        head=newnode;
}

void InsertEnd()
{
        NODE *newnode,*p;
        newnode=Getnode();
        if(head==NULL)
                head=newnode;
        else
        {
                for(p=head;p->next!=NULL;p=p->next); //note ;
                p->next=newnode;
        }
}

void InsertPos()
{
        int pos;
        printf("\n\n\tEnter the Inserting Position: ");
        scanf("%d",&pos);
        if(pos<1||pos>count+1)
                printf("\n\tEnter Valid Position...");
        else
        {
                NODE *p,*q;
                int i;
                for(i=1,p=head;i<pos&&p!=NULL;q=p,p=p->next,i++);//note ;
                if(i==1)
                        InsertFront();
                else if(i==count+1)
                        InsertEnd();
                else
                {
                        NODE *newnode;
                        newnode=Getnode();
                        newnode->next=q->next;
                        q->next=newnode;
                }
         }
}
.....
```

```
void Insert()
{
        int ch;
        printf("\n\t1.Insert at Front...");
        printf("\n\t2.Insert at End...");
        printf("\n\t3.Insert at Given Position...");
        printf("\n\n\tEnter Your Choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
                case 1: InsertFront(); break;
                case 2: InsertEnd(); break;
                case 3: InsertPos(); break;
                default: printf("\n\n\tEnter Proper Choice....");
        }
}

// ****************** Sub-Program B ***********************

void DeleteId()
{
            if(head==NULL)
                printf("\n\n\tList is Empty...");
            else
            {
                NODE *p,*q;
                int id;
                printf("\n\n\tEnter the Student Id to be Deleted: ");
                scanf("%d",&id);
                p=head;
                for(p=head;p->id!=id&&p!=NULL;q=p,p=p->next); //note ;
                if(p==NULL)
                        printf("\n\n\t Student with Id %d does not exist...",id);
                else
                {
                        printf("\n\n\tThe Fallowing Student is Deleted:");
                        printf("\n\n\t\t\tName: %s",p->name);
                        printf("\n\t\t\tId    : %d",p->id);
                        printf("\n\t\t\tSem   : %d",p->sem);
                        if(p==head)
                                head=head->next;
```

```
                        else
                                q->next=p->next;
                        free(p);
                        count--;
                }
        }
}
// ******************* Sub-Program C **********************
void UpdateId()
{
        if(head==NULL)
                printf("\n\n\tList is Empty...");
        else
        {
                NODE *p,*newnode;
                int id;
                printf("\n\n\tEnter the Student Id to be Updated: ");
                scanf("%d",&id);
                p=head;
                for(p=head;p->id!=id&&p!=NULL;p=p->next); //note ;
                if(p==NULL)
                        printf("\n\n\tStudent with Id %d does not exist...",id);
                else
                {
                        printf("\n\n\tThe Fallowing Student is to be Updated:");
                        printf("\n\n\t\t\tName  : %s",p->name);
                        printf("\n\t\t\tId    : %d",p->id);
                        printf("\n\t\t\tSem   : %d",p->sem);
                        printf("\n\tEnter Id    : "); scanf("%d",&p->id);
                        printf("\n\n\tEnter Name: "); scanf("%s",p->name);
                        printf("\n\tEnter Sem   : "); scanf("%d",&p->sem);
                }
        }
}

// ******************* Sub-Program D **********************
void Display()
{
        int i;
        if(head==NULL)
                printf("\n\n\tList is Empty...");
        else
        {
                NODE *p;
                int i;
                for(i=1,p=head;p!=NULL;p=p->next,i++)
```

```
                {
                        printf("\n\nStudent-%d Details:",i);
                        printf("\n\t\t\tName: %s",p->name);
                        printf("\n\t\t\tId  : %d",p->id);
                        printf("\n\t\t\tSem : %d",p->sem);
                 }
                 printf("\n\n\tTotal Number of Students are: %d",count);
        }
}
void main()
{
        int choice;
        clrscr();
        head=NULL;
        while(1)
        {
                printf("\n\n\t1.Insert..\t2.Delete..\t3.Search..\t4.Display..\t5.Exit..");
                printf("\n\n\n\tEnter Your Choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: Insert(); break;
                        case 2: DeleteId(); break;
                        case 3: UpdateId(); break;
                        case 4: Display(); break;
                        case 5: exit(0);
                        default: printf("\n\n\tEnter Proper Choice...");
                }
        }
}
```

**OUTPUT**

```
1.Insert..    2.Delete..    3.Search..    4.Display..    5.Exit..

Enter Your Choice: 1

1.Insert at Front...
2.Insert at End...
3.Insert at Given Position...

Enter Your Choice: 1
```

Enter Student Id: 100

Enter Name : raju

Enter Semester : 3

1.Insert.. 2.Delete.. 3.Search.. 4.Display.. 5.Exit..

Enter Your Choice: 1

1.Insert at Front...
2.Insert at End...
3.Insert at Given Position...

Enter Your Choice: 1

Enter Student Id: 101

Enter Name : nikhil

Enter Semester : 5

1.Insert.. 2.Delete.. 3.Search.. 4.Display.. 5.Exit..

Enter Your Choice: 4

Student-1 Details:
Name: nikhil
Id : 101
Sem : 5

Student-2 Details:
Name: raju
Id : 100
Sem : 5

Total Number of Students are: 2

1.Insert.. 2.Delete.. 3.Search.. 4.Display.. 5.Exit..

Enter Your Choice: 3

Enter the Student Id to be Updated: 100

The Following Student is to be Updated:

Name : raju
Id : 100
Sem : 3
Enter Id : 100

Enter Name: raju

Enter Sem : 5

1.Insert.. 2.Delete.. 3.Search.. 4.Display.. 5.Exit..

Enter Your Choice: 4

Student-1 Details:
Name: nikhil
Id : 101
Sem : 5

Student-2 Details:
Name: raju
Id : 100
Sem : 5

Total Number of Students are: 2

1.Insert.. 2.Delete.. 3.Search.. 4.Display.. 5.Exit..

Enter Your Choice: 3

Enter the Student Id to be Updated: 150

Student with Id 150 does not exist...

1.Insert.. 2.Delete.. 3.Search.. 4.Display.. 5.Exit..

Enter Your Choice: 2

Enter the Student Id to be Deleted: 101

The Fallowing Student is Deleted:

Name: nikhil
Id : 101
Sem : 5

1.Insert.. 2.Delete.. 3.Search.. 4.Display.. 5.Exit..

Enter Your Choice: 4

Student-1 Details:
Name: raju
Id : 100
Sem : 5

Total Number of Students are: 1

1.Insert.. 2.Delete.. 3.Search.. 4.Display.. 5.Exit..

Enter Your Choice: 5

10. Write a C program using dynamic variables and pointers to construct a stack of integers using singly linked list and to perform the following operations.
    a) Push b) Pop c) Display

    The program should print appropriate messages for stack overflow & stack empty.

**ALGORITHM FOR STACK USING LINKED LIST**

MAIN FUNCTION()

S1: Initialize stack using linked list data structure, where stack is called the external pointer to the list.
S2: Consider a variable called count and initialize it to zero.
S3: Read the operation to be performed from the keyboard.
S4:If the operation specified is push to go to the push function , if the operation specified is pop function go to pop function else if the operation specified is display go to display function else if the operation specified is exit go to step5.
S5: Stop

PUSH FUNCTION()

S1: Check if the stack is full, if yes quit with suitable error message else go to step2.
S2: Create a node dynamically called new node containing data field and the link field and also assign the link field to a null value.
S3: Insert the data to be entered by the user to the data fields of the new node.
S4: Assign the stack pointer to the link field of the new node.
S5: Assign new node address to the stack.
S6: Increment count by one.
S7: Return to the main program

POP FUNCTION()

S1: Check if the stack empty if yes quit and display stack underflow else go to step2.
S2: Consider a temporary called p and assign the stack address to p.
S3: Store the link address of stack to stack
S4: Free P or deallocate P
S5: Decrement the count variable by one.
S6: Return to the main program

DISPLAY FUNCTION()

S1: Check if the stcak is empty, if yes quit and display stack underflow else go to step2.
S2: Consider a temporary node called P and assin the address of stack to P.
S3: Repeat step4 to step5 unless P is null.
S4: Display the contents f the data field of the node P.
S5: Assign the link address of P to P
S6: Return to the main program

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 3

struct Node
{
        int info;
        struct Node *next;
};
typedef struct Node NODE;

NODE *Stack;
int count;

void Push()
{
        if(count==MAX)
                printf("\n\n\tStack OverFlow...");
        else
        {
                NODE *newnode;
                newnode=(NODE *)malloc(sizeof(NODE));
                printf("\n\n\tEnter an Element to Push: ");
                scanf("%d",&newnode->info);
```

```
                newnode->next=Stack;
                Stack=newnode;
                count++;
        }
}

void Pop()
{
        if(Stack==NULL)
                printf("\n\n\tStack UnderFlow...");
        else
        {
                NODE *p;
                printf("\n\n\tElement Popped is: %d",Stack->info);
                p=Stack;
                Stack=Stack->next;
                free(p);
                count--;
        }
}

void Display()
{
        int i;
        if(Stack==NULL)
                printf("\n\n\tStack UnderFlow...");
        else
        {
                NODE *p;
                printf("\n\nElements of the Stack are: ");
                for(p=Stack;p!=NULL;p=p->next);
                printf("\n\t\t\t\t%d",p->info);
                printf("\n\n\ttotal Elements are: %d",count);
        }
}
void main()
{
        int choice;
        clrscr();
        Stack=NULL;
        while(1)
        {
                printf("\n\n\t1.Push...\t2.Pop...\t3.Display...\t4.Exit...");
                printf("\n\n\n\tEnter Your Choice: ");
                scanf("%d",&choice);
                witch(choice)
```

```
                {
                        case 1: Push(); break;
                        case 2: Pop(); break;
                        case 3: Display(); break;
                        case 4: exit(0);
                        default: printf("\n\n\tEnter Proper Choice...");
                }
        }
}
```

**OUTPUT**

```
1.Push...    2.Pop...    3.Display...    4.Exit...

Enter Your Choice: 1

Enter an Element to Push: 12

1.Push...    2.Pop...    3.Display...    4.Exit...

Enter Your Choice: 1

Enter an Element to Push: 23

1.Push...    2.Pop...    3.Display...    4.Exit...

Enter Your Choice: 1

Enter an Element to Push: 45

1.Push...    2.Pop...    3.Display...    4.Exit...

Enter Your Choice: 1

Stack OverFlow...
```

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 3

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 3

Elements of the Stack are:
45
23
12

total Elements are: 3

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 2

Element Popped is: 45

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 2

Element Popped is: 23

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 2

Element Popped is: 12

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 2

Stack UnderFlow...

1.Push... 2.Pop... 3.Display... 4.Exit...

Enter Your Choice: 4

11. Write a C program using dynamic variables and pointers to construct a queue of integers using singly linked list and to perform the following operations.
a) Insert b) Delete c) Display
The program should print appropriate messages for Queue overflow & Queue empty.

**ALGORITHM FOR QUEUES USING SINGLY LINKED LIST PROGRAM**

MAIN FUNCTION()

S1: Declare the data structure to represent a dynamic node with data field to store information and a link field to hold address of the next node.
S2: Declare an identifier queue as an external pointer to list of nodes initialized to Null
S3: Consider a variable called count and initialize it to zero .
S4: Read the operations to be performed from the keyboard .
S5: If the operation specified is insert go to the insert function , if the operation specified is delete go to the delete function, if the operation specified is display go to the display function, if the operation specified is exit go to step6.
S6: Stop.

INSERT FUNCTION()

S1: Check if the queue is full , if yes display queue overflow and quit , else go to step2 .
S2: Create a node dynamically called newnode containing a data field and a link field and assign the link field to Null value .
S3: Insert the data to be entered by the user to the data field of the newnode .
S4: Scan the queue for the occurrence of the last node in queue .
S5: Assign the newnode address to the link of the last node in queue.
S6: Increment the count by one .
S7: Return to the main program.

DELETE FUNCTION()

S1 : Check if the queue is empty , if yes display queue underflow condition and quit , else go to step2.
S2: Consider a temporary node called p .
S3: Assign the queue address to p .
S4: Store the queue link address to the queue .
S5: Free the node p .
S6: Decrement the count by one .
S7: Return to the main program

DISPLAY FUNCTION()

S1: Check if the queue is empty , if yes display queue underflow condition and quit , else go to step 2.
S2: Scan for each node from queue and display the contents of info part of each node till the end of list each reached .
S3: Return to the main program.

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 5

struct Node
{
        int info;
        struct Node *next;
};
typedef struct Node NODE;

NODE *Queue;
int count;
void Insert()
{
```

```
        if(count==MAX)
                printf("\n\n\tQueue OverFlow...");
        else
        {
                NODE *newnode;
                newnode=(NODE *)malloc(sizeof(NODE));
                printf("\n\n\tEnter an Element to Insert: ");
                scanf("%d",&newnode->info);
                newnode->next=Queue;
                Queue=newnode;
                count++;
        }
}
void Remove()
{
        if(Queue==NULL)
                printf("\n\n\tQueue UnderFlow...");
        else
        {
                NODE *p,*q;
                for(p=Queue;p->next!=NULL;q=p,p=p->next); //note ;
                printf("\n\n\tElement Removed is: %d",p->info);
                if(p==Queue)
                        Queue=NULL;
                else
                        q->next=NULL;
                free(p);
                count--;
         }
}
void Display()
{
         int i;
         if(Queue==NULL)
                printf("\n\n\tQueue UnderFlow...");
         else
         {
                NODE *p;
                printf("\n\nElements of the Queue are: ");
                for(p=Queue;p!=NULL;p=p->next)
                        printf("\n\t\t\t\t%d",p->info);
                printf("\n\n\ttotal Elements are: %d",count);
         }
}

void main()
```

```
{
        int choice;
        clrscr();
        Queue=NULL;
        while(1)
        {
                printf("\n\n\t1.Insert...\t2.Remove...\t3.Display...\t4.Exit...");
                printf("\n\n\n\tEnter Your Choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: Insert(); break;
                        case 2: Remove(); break;
                        case 3: Display(); break;
                        case 4: exit(0);
                        default: printf("\n\n\tEnter Proper Choice...");
                }
        }
}
```

**OUTPUT**

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert: 11

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert: 22

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Enter an Element to Insert: 33

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 1

Queue OverFlow...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 3

33
22
11

total Elements are: 3

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

Element Removed is: 11

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

Element Removed is: 22

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

Element Removed is: 33

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

Queue UnderFlow...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 2

Queue UnderFlow...

1.Insert... 2.Remove... 3.Display... 4.Exit...

Enter Your Choice: 4

12. Write a C program to support the following operations on a doubly linked list where each node consists of integers.
    a) Create a doubly linked list by adding each node at the front.
    b) Insert a new node to the left of the node whose key value is read as an input.
    c) Delete the node of a given data, if it is found, otherwise display appropriate message.
    d) Display the contents of the list.

**ALGORITHM FOR DOUBLY LINKED LIST**

MAIN FUNCTION()

S1: Initialize a list using doubly linked list data structure where head is called as the start node.
S2: Read the operation to be performed from the keyboard.
S3: If the operation specified is insert front go to insert front function,
if the operation specified is delete then go to the delete function
else if the operation specified is to display go to the display function,
else if the operation specified is exit go to S4.
S4: Stop.

INSERT FRONT FUNCTION()

S1: Create a node called new node containing a data field and a right link field and left link field and also assign both the right link and left link to null value.
S2: Insert the data to be entered by the user to the data field of the new node.
S3: Assign the head address to the new node address to the left link of head.
S4: Reassign the head address with the new node address.
S5: Return to the main program.

INSERT TO THE LEFT OF THE KEY()

S1: Check if the list is empty if yes display list is empty and qui else go to step 2.
S2: Create a node called new node containing data field and two link field namely rightly link and left link respectively. Assign the right link and left links to null value.
S3: Insert the data to be entered by the user to the data field of new node.
S4: Read the value of a key from keyboard.
S5: Scan the list to reach the key position in the list if the key is not found then display key not found and quit else go to step6.
S6: Check if key is the first element in list of yes then call Insert front function else go to step7
S7: Consider temporary node p and assign the node at key value to p.
S8: Assign left pointer to p to left pointer of new node and assign the address of p to right link of new node.
S9: Assign the new node address to right link of p's preview node.
S10: Assign the new node's address to left link of p.
S11: Return to the main program

DELETE FUNCTION()

S1: Check if the list is empty, if yes then display list is empty and quit else go to step2.
S2: Read the element to be deleted from the list.
S3: Scan the occurrence of the element in the list. If the element not found then display element not found and quit else go to step4.
S4: Assign the node at which element was found to a variable called p.
S5: Assign P's right link address to the P's previous node right link.
S6: Assign P's left link address to the P's next node left link.
S7: De allocate P's memory.

S8: Return to the main program.

DISPLAY FUNCTION()

S1: Check if the list is empty, if yes then display list is empty and quit else go to step step2.
S2: Scan each and every node from the head node onwards and display the info part of every node scanned.
S3: Return to the main program

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 5

struct Node
{
        int info;
        struct Node *right;
        struct Node *left;
};
typedef struct Node NODE;
NODE *head;
int count;

NODE *Getnode()
{
        NODE *newnode;
        newnode=(NODE *)malloc(sizeof(NODE));
        newnode->right=newnode->left=NULL;
        printf("\n\tEnter The Number: ");
        scanf("%d",&newnode->info);
        count++;
        return newnode;
}



// ****************** Sub-Program A **********************

void InsertFront()
{
        NODE *newnode;
        newnode=Getnode();
```

```
        newnode->right=head;
        head->left=newnode;
        head=newnode;
}

// ****************** Sub-Program B **********************

void InsertLeftofKey()
{
        if(head==NULL)
                printf("\n\n\tList is Empty...");
        else
        {
                int key;
                NODE *p;
                printf("\n\nEnter Key element to insertnew elemnt to its left: ");
                scanf("%d",&key);
                for(p=head;p!=NULL&&p->info!=key;p=p->right); //note ;
                if(p==NULL)
                        printf("\n\n\tKey %d does not exist...",key);
                else if(p==head)
                        InsertFront();
                else
                {
                        NODE *newnode;
                        newnode=Getnode();
                        newnode->left=p->left;
                        newnode->right=p;
                        p->left->right=newnode;
                        p->left=newnode;
                }
        }
}

// ****************** Sub-Program C **********************
void Delete()
{
        if(head==NULL)
                printf("\n\n\tList is Empty...");
```

```
        else
        {
                NODE *p;
                int info;
                printf("\n\n\tEnter the Element to Delete: ");
                scanf("%d",&info);
                p=head;
                for(p=head;p->info!=info&&p!=NULL;p=p->right); //note ;
                if(p==NULL)
                        printf("\n\n\tElement %d does not exist...",info);
                else
                {
                        printf("\n\n\tElement %d is Deleted...",info);
                        if(p==head)
                        {
                                head=head->right;
                                head->left=NULL;
                        }
                        else
                        {
                                p->left->right=p->right;
                                p->right->left=p->left;
                        }
                        free(p);
                        count--;
                 }
        }
}
// ******************* Sub-Program D ************************
void Display()
{
        int i;
        if(head==NULL)
                printf("\n\n\tList is Empty...");
        else
        {
                NODE *p;
                printf("\n\n\tElements of the List are:");
                for(p=head;p!=NULL;p=p->right)
                        printf(" %d",p->info);
                 printf("\n\n\tTotal Number of Elements are: %d",count);
        }
}
void main()
{
        int choice;
```

```
        clrscr();
        head=NULL;
        while(1)
        {
                printf("\n\n\t1.Create...\t2.Insert...\t3.Delete...\t4.Display...\t5.Exit..");
                printf("\n\n\n\tEnter Your Choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: InsertFront(); break;
                        case 2: InsertLeftofKey(); break;
                        case 3: Delete(); break;
                        case 4: Display(); break;
                        case 5: exit(0);
                        default: printf("\n\n\tEnter Proper Choice...");
                }
        }
}
```

**OUTPUT**

1.Create... 2.Insert... 3.Delete... 4.Display... 5.Exit..

Enter Your Choice: 1

Enter The Number: 12

1.Create... 2.Insert... 3.Delete... 4.Display... 5.Exit..

Enter Your Choice: 1

Enter The Number: 14

1.Create... 2.Insert... 3.Delete... 4.Display... 5.Exit..

Enter Your Choice: 2

Enter Key element to insertnew elemnt to its left: 14

Enter The Number: 66

1.Create... 2.Insert... 3.Delete... 4.Display... 5.Exit..

Enter Your Choice: 4

Elements of the List are: 66 14 12

Total Number of Elements are: 3

1.Create... 2.Insert... 3.Delete... 4.Display... 5.Exit..

Enter Your Choice: 3

Enter the Element to Delete: 14

Element 14 is Deleted...

1.Create... 2.Insert... 3.Delete... 4.Display... 5.Exit..

Enter Your Choice: 4

Elements of the List are: 66 12

Total Number of Elements are: 2

1.Create... 2.Insert... 3.Delete... 4.Display... 5.Exit..

Enter Your Choice: 5

13. Write a C Program
    a. To construct a binary search tree of integers.
    b. To traverse the tree using all the methods i.e., inorder, preorder and postorder.
    c. To display the elements in the tree.

**ALGORITHM FOR TREES**

MAIN FUNCTION()

S1: Initialize a list called tree using tree data structures , consider root as the start node .
S2: Read the operation from the keyboard , if the operation says preorder display then go to the preorder display function else if the operation specifies post order then go to the post order display function else if the operation specifies in order then go to the in order display function else if the operation specified is exit then go to step step3 .
S3: Return to the main program

INORDER FUNCTION()

S1: Traverse the left subtree by calling Inorder function recursively.
S2: Visit root and print root information.
S3: Traverse the right subtree by calling Inorder function recursively.
S4: Return to the main program

PREORDER DISPLAY FUNCTON()

S1: Visit root and print root information .
S2: Traverse the left subtree by calling Preorder function recursively.
S3: Traverse the right subtree by calling Preorder function recursively.
S4: Return to the main program

POST ORDER DISPLAY FUNCTION()
S1: Traverse the left subtree by calling Postorder function recursively.
S2: Traverse the right subtree by calling Postorder fuction reecursively.
S3: Visit root and print root information.
S4: Return to the main program

INSERTION FUNCTION()

S1: Create a node called newnode with a data field and a left and the right link .
S2: Insert the value entered from the user to the data field of the newnode and assign the right and left links to the null value .
S3: Check if the tree is empty, if yes then newnode is the root node and goto step9 else goto step4.
S4: Consider a temporary node p and assign it with root .
S5: Repeat S6 and S7 till p becomes Null .
S6: Consider a temporary node q and assign it with p .
S7: Check if newnode data is greater then p node data, if yes proceed towards right subtreee else proceed towards left subtree.

S8: If new node data is less than q node data insert newnode as left child of q node else insert newnode as right child of q node .
S9: Return to the main program.

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

typedef struct Node
{
        int info;
        struct Node *left;
        struct Node *right;
}NODE;

NODE *root;

void Inorder(NODE *p)
{
        if(p!=NULL)
        {
                Inorder(p->left);
                printf(" %d",p->info);
                Inorder(p->right);
        }
}

void Preorder(NODE *p)
{
        if(p!=NULL)
        {
                printf(" %d",p->info);
                Preorder(p->left);
                Preorder(p->right);
        }
}

void Postorder(NODE *p)
{
        if(p!=NULL)
        {
```

```
                Postorder(p->left);
                Postorder(p->right);
                printf(" %d",p->info);
        }
}



void Insert()
{
        NODE *newnode;
        newnode=(NODE *)malloc(sizeof(NODE));
        newnode->left=newnode->right=NULL;
        printf("\n\n\tEnter an Elemnet to Insert: ");
        scanf("%d",&newnode->info);
        if(root==NULL)
                root=newnode;
        else
        {
                NODE *p,*q;
                p=root;
                while(p!=NULL)
                {
                        q=p;
                        if(newnode->info>p->info)
                                p=p->right;
                        else
                                p=p->left;
                }
                if(newnode->info>q->info)
                        q->right=newnode;
                else
                        q->left=newnode;
        }
}

void Display()
{
        if(root==NULL)
                printf("\n\n\tNo Nodes in the Tree....");
        else
        {
                printf("\n\n\tPreoredr Traversal  :"); Preorder(root);
                printf("\n\n\tInoredr Traversal   :"); Inorder(root);
                printf("\n\n\tPostoredr Traversal :"); Postorder(root);
        }
```

```
}
```

```
void main()
{
        int choice;
        clrscr();
        root=NULL;
        while(1)
        {
                printf("\n\n\n\t1.Insert...\t2.Display...\t3.Exit...");
                printf("\n\n\n\tEnter Your Choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: Insert(); break;
                        case 2: Display(); break;
                        case 3: exit(0);
                        default: printf("\n\n\tEnter Proper Choice...");
                 }
        }
}
```

**OUTPUT**

```
1.Insert...     2.Display...    3.Exit...

Enter Your Choice: 1

Enter an Element to Insert: 10

1.Insert...     2.Display...    3.Exit...
```

Enter Your Choice: 1

Enter an Element to Insert: 8

1.Insert... 2.Display... 3.Exit...

Enter Your Choice: 1

Enter an Element to Insert: 9

1.Insert... 2.Display... 3.Exit...

Enter Your Choice: 2

Preoredr Traversal : 10 8 9

Inoredr Traversal : 8 9 10

Postoredr Traversal : 9 8 10

1.Insert... 2.Display... 3.Exit...

Enter Your Choice: 3

14a. Write recursive C programs for
a) Searching an element on a given list of integers using the Binary search method.

**ALGORITHM FOR TOWERS OF HANOI**

MAIN FUNCTION()

S1: Read No of disks called n from keyboard.
S2: Check if n is not zero or a negative no. if yes display suitable message else go to step3.
S3: Call tower of Hanoi function with n as parameter,
S4: Stop

TOWERS OF HANOI FUNCTION TO MOVE DISKS FROM A TO C USING B()

S1: If n is equal to 1 then move the single disk from A to C and stop
S2: Move the top n-1 disks from A to B using c as auxillary.
S3: Move the remaining disk from A to C.
S4: Move the n-1 disks from B to C using as auxillary.

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#define SIZE 10

int a[SIZE];
int key,n;

int BinarySearch(int low,int high)
{
        if(low>high)
                return -1;
        else
```

```
        {
                int mid=(low+high)/2;
                if(key==a[mid])
                        return mid;
                else if(key<a[mid])
                        return BinarySearch(low,mid-1);
                else
                        return BinarySearch(mid+1,high);
        }
}



void Sort()
{
        int i,j;
        for(i=0;i<n-1;i++)
        {
                for(j=i+1;j<n;j++)
                if(a[i]>a[j])
                {
                        int t=a[i];
                        a[i]=a[j];
                        a[j]=t;
                }
        }
}


void main()
{
        int position,i;
        clrscr();
        printf("\n\n\tEnter Number of elements: ");
        scanf("%d",&n);
        printf("\n\n\tEnter %d Numbers: ",n);
        for(i=0;i<n; i++)
        scanf("%d",&a[i]);
        printf("\n\n\tEnter the Key element to be searched: ");
        scanf("%d",&key);
        Sort();
        position=BinarySearch(0,n-1);
        if(position==-1)
                printf("\n\n\tSorry Element does not exist....");
        else
```

```
            printf("\n\n\tElement present and the position is: %d",position+1);
        getch();
}
```

**OUTPUT**

Enter Number of elements: 5

Enter 5 Numbers: 1 3 45 67 89

Enter the Key element to be searched: 45

Element present and the position is: 3

14 Write recursive C programs for
b) Solving the Towers of Honai problem.

**ALGORITHM FOR BINARY SEARCH**

MAIN FUNCTION()

S1: Initialize the list called a using array data structure.
S2: Read the total no. of elements called n that need to entered to the list from keyboard.
S3: Read n different values to the list from user.
S4 Check if the element are sorted, if yes go to s5 else call the sort function.
S5: Read the key to be searched from keyboard.
S6: Call binary search function with key as input parameters recursively.
S7 :Stop.

SORT FUNCTION()

S1: Repeat s2 to s3 for n cycles.
S2: Compare each consecutive pair of elements.
S3: If the first element is greater than the second the swap their respective position.
S4: Return to the main program

BINARY SEARCH FUNCTION()

S1: Consider three variables called low, high and mid to mark initial, middle and final positions of list.
S2: Repeat the steps s3 to s5 unless low is less than or equal high.
S3: Find the middle position of the array by taking the average of the initial and the final position of the array.
S4: Compare the middle position element of the list with the key element.

S5: If both are equal then display key found set flag and quit else if key is less then middle position element search in the first phase of the array by shifting high to mid-1 and call the binary search function else search in second phase by shifting low to mid+1 position and call binary search function.
S6: Check if flag is zero then display search not successful and quit.
S7: Return to the main program.

**PROGRAM**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

void Tower(int n, char frompeg, char auxpeg,char topeg)
{
        if(n==1)
                printf("\n\t\tDisk is moved from %c to %c",frompeg, topeg);
        else
        {
                Tower(n-1,frompeg,topeg,auxpeg);
                 printf("\n\t\tDisk is moved from %c to %c",frompeg, topeg);
                Tower(n-1,auxpeg,topeg,frompeg);
        }
}

void main()
{
        int n;
        clrscr();
        printf("\n\n\tEnter Number of Disk: ");
        scanf("%d",&n);
        if(n<1)
```

```
                printf("\n\n\tEnter Valid Number of Disk...");
        else
        {
                Tower(n,'A','B','C');
                printf("\n\n\n\t\tTotal Number of moves are: %d",(int) pow(2,n)-1);
        }
        getch();
}
```

**OUTPUT**

```
 Enter Number of Disk: 3
Disk is moved from A to C
Disk is moved from A to B
Disk is moved from C to A
Disk is moved from A to C
Disk is moved from B to C
Disk is moved from B to A
Disk is moved from C to B

Total Number of moves are: 7
```