DATA STRUCTURES LAB MANUAL -P22CSL306

Prof H P Ramyashree

ASSISTANT PROFESSOR ,Computer Science & Engineering Department

Program 1:

Create a structure DISTANCE with data members kms and meters.
Implement a program using function to perform addition and subtraction on two distances by passing pointer to a structure.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Distance
{
  int km;
  int meters;
}D;
D d1,d2,d3,d4;
void Read(D *d)
{
   printf("\nEnter Distance in Km and meter:\n");
    scanf("%d%d",&d->km,&d->meters);
}
void Display(D d)
{
   printf("\n %d km %d meters\n",d.km,d.meters);
}
D Subtract(D *d1 , D *d2, D *d3)
{

   if(d1->km>d2->km)
   {
      d3->km=d1->km-d2->km;
      d3->meters=d1->meters-d2->meters;
   }
   else if(d1->km<d2->km)
   {
      d3->km=d2->km-d1->km;
      d3->meters=d2->meters-d1->meters;
   }
   else
   {
      d3->km=0;
      if(d1->meters>d2->meters)
          d3->meters=d1->meters-d2->meters;
      else
          d3->meters=d2->meters-d1->meters;
   }
```

```
    if(d3->meters<0)
    {
       d3->meters+=1000;
       --(d3->km);
    }
}

void Add (D * d1 , D *d2 , D *d3)
{

    d3->km=d1->km+d2->km;
    d3->meters=d1->meters+d2->meters;
    if(d3->meters>=1000)
    {
       d3->km+=(d3->meters/1000);
       d3->meters%=1000;
    }

}

int main()
{
    int ch;
    for(;;)
    {
    printf("\n\tMENU\n");
    printf("1.To read Distances\n");
    printf("2.To Add 2 Distances\n");
    printf("3.To Subtract 2 Distances\n");
    printf("4.To Display Distances Read\n");
    printf("5.To Display Sum Result\n");
    printf("6.To Display Subtraction  Result\n");
    printf("7.To Exit\n");
    printf("Enter your choice :\n");
    scanf("%d",&ch);
    switch (ch)
    {
 case 1:
     printf("\nEnter the first Distance :");
     Read(&d1);
     printf("\nEnter the Second Distance :");
```

```
    Read(&d2);
    break;
 case 2:
    Add(&d1,&d2,&d3);
    break;
 case 3:
   Subtract(&d1,&d2,&d4);
    break;
 case 4:
    printf("\nfirst Distance :");
    Display(d1);
    printf("\nEnter the Second Distance :");
    Display(d2);
    break;
  case 5:
    printf("\nSum of Distance :");
    Display(d3);
    break;
  case 6:
    printf("\nDifference of Distance :");
    Display(d4);
    break;
   case 7:
   exit(0);
   default:
      printf("\nInvalid Choice ");
   }
   }
   return 0;
}
```

Output:

```
 MENU
1.To read Distances
2.To Add 2 Distances
3.To Subtract 2 Distances
4.To Display Distances Read
5.To Display Sum Result
6.To Display Subtraction  Result
7.To Exit
```

Enter your choice :
1

Enter the first Distance :
Enter Distance in Km and meter:
50
2

Enter the Second Distance :
Enter Distance in Km and meter:
49
60

MENU
1.To read Distances
2.To Add 2 Distances
3.To Subtract 2 Distances
4.To Display Distances Read
5.To Display Sum Result
6.To Display Subtraction Result
7.To Exit
Enter your choice :
2

MENU
1.To read Distances
2.To Add 2 Distances
3.To Subtract 2 Distances
4.To Display Distances Read
5.To Display Sum Result
6.To Display Subtraction Result
7.To Exit
Enter your choice :
3

MENU
1.To read Distances
2.To Add 2 Distances
3.To Subtract 2 Distances
4.To Display Distances Read
5.To Display Sum Result

6.To Display Subtraction Result
7.To Exit
Enter your choice :
4

first Distance :
 50 km 2 meters

Enter the Second Distance :
 49 km 60 meters

MENU
1.To read Distances
2.To Add 2 Distances
3.To Subtract 2 Distances
4.To Display Distances Read
5.To Display Sum Result
6.To Display Subtraction Result
7.To Exit
Enter your choice :
5

Sum of Distance :
 99 km 62 meters

MENU
1.To read Distances
2.To Add 2 Distances
3.To Subtract 2 Distances
4.To Display Distances Read
5.To Display Sum Result
6.To Display Subtraction Result
7.To Exit
Enter your choice :
6

Difference of Distance :
 0 km 942 meters

MENU
1.To read Distances

2.To Add 2 Distances
3.To Subtract 2 Distances
4.To Display Distances Read
5.To Display Sum Result
6.To Display Subtraction Result
7.To Exit
Enter your choice :
7
Also check for:
Distance 1: 50 km 60 m Distance 2: 50 km 2 m
Ans: 58m

Program 2:

Implement a menu driven program to perform the following operations on Singly Linked List.
i. Create SLL of 'n' nodes of integers (insert front/rear)
ii. Delete the node with specified integer from the list with appropriate message.
iii. Display the contents of the SLL.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node
{
   int data;
   struct Node *link;
}*NODE;

NODE getNode()
{
NODE    NewNode=(NODE)malloc(sizeof(struct Node));
   if(!NewNode)
      printf("Memory Allocation Failed\n");
   printf("Enter the integer Data:\n");
   scanf("%d",&NewNode->data);
   NewNode->link=NULL;
   return NewNode;
}

NODE insertF(NODE front , NODE newNode)
{
   newNode->link=front;
   return newNode;
}

NODE insertR(NODE front , NODE newNode)
{
   if(front==NULL)
      return newNode;
   NODE temp=front;
   while(temp->link!=NULL)
   {
      temp=temp->link;
   }
      temp->link=newNode;
   return front;
```

```
}

NODE Delete(NODE front)
{
    if(front==NULL)
    {
      printf("\nList is Empty");
      return front;
    }
    int ele;
    printf("\nEnter element to delete:\n");
    scanf("%d",&ele);
   NODE temp=front;
   NODE prev=NULL;
   if(temp->data==ele)
   {
      front=front->link;
      free(temp);
      return front;
   }
   while(temp!=NULL)
   {
      if(temp->data==ele)
      break;
      prev=temp;
      temp=temp->link;
   }
   if(temp==NULL)
      printf("\nElement Not found \n");
   else
   {
      prev->link=temp->link;
      free(temp);
   }

   return front;
}
void Display(NODE front)
{
   if(front==NULL)
    {
```

```
        printf("\nList is Empty");
     }
    else{
   while(front!=NULL)
   {
     printf("\n%d",front->data);
     front=front->link;
   }
    }

}
int main()
{
    int ch;
    NODE front=NULL;
    for(;;)
    {
    printf("\n\tMenu\n");
    printf("1: To insert Front\n");
    printf("2: To insert Rear\n");
    printf("3: To Delete Specified Element\n");
    printf("4: To Display\n");
    printf("Enter your Choice :\n");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
    front=insertF(front , getNode());
    break;
  case 2:
    front=insertR(front , getNode());
    break;
    case 3:
    front=Delete(front);
    break;
    case 4:
    Display(front);
    break;
    case 5:
        exit(0);
    default:
```

```
        printf("\nInvalid Choice\n");

        }
    }
    return 0;
}
```

Output:

```
        Menu
1: To insert Front
2: To insert Rear
3: To Delete Specified Element
4: To Display
Enter your Choice :
1
Enter the integer Data:
1

        Menu
1: To insert Front
2: To insert Rear
3: To Delete Specified Element
4: To Display
Enter your Choice :
1
Enter the integer Data:
0

        Menu
1: To insert Front
2: To insert Rear
3: To Delete Specified Element
4: To Display
Enter your Choice :
2
Enter the integer Data:
2

        Menu
1: To insert Front
```

```
2: To insert Rear
3: To Delete Specified Element
4: To Display
Enter your Choice :
4

0
1
2
      Menu
1: To insert Front
2: To insert Rear
3: To Delete Specified Element
4: To Display
Enter your Choice :
3

Enter element to delete:
1

      Menu
1: To insert Front
2: To insert Rear
3: To Delete Specified Element
4: To Display
Enter your Choice :
4

0
2
      Menu
1: To insert Front
2: To insert Rear
3: To Delete Specified Element
4: To Display
Enter your Choice :
5

Process returned 0 (0x0)   execution time : 48.377 s
Press any key to continue.
```

Program 3:

Implement a menu driven Program for the following operations on Doubly Linked List (DLL) of Library Data with the fields: BOOK_ID, BOOK_TITLE, AUTHOR, EDITION
i. Create a DLL of 'N' books (Insert front/rear).
ii. Count the number of nodes in the DLL.
iii. Delete the node at front/rear
iv. Display the contents of DLL

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

typedef struct Node {
   char BT[100];
   char AN[100];
   int Edi;
   struct Node *Llink;
   struct Node *Rlink;
} Node;

typedef Node* NODE;

NODE getNode() {
   NODE NewNode = (NODE)malloc(sizeof(Node));
   if (!NewNode) {
      printf("Memory Allocation Failed\n");
      exit(EXIT_FAILURE);
   }
   printf("Enter the Book Name:\n");
   scanf(" %99[^\n]", NewNode->BT);
   printf("Enter the author Name:\n");
   scanf(" %99[^\n]", NewNode->AN);
   printf("Enter the Book Edition:\n");
   scanf("%d", &NewNode->Edi);
   NewNode->Llink = NULL;
   NewNode->Rlink = NULL;
   return NewNode;
}

NODE insertF(NODE front, NODE newNode) {
   if (front == NULL) {
      return newNode;
   }
   newNode->Rlink = front;
```

```
    return newNode;
}

NODE insertR(NODE front, NODE newNode) {
    if (front == NULL) {
        return newNode;
    }
    NODE temp=front;
    while(temp->Rlink!=NULL)
    {
        temp=temp->Rlink;
    }
    temp->Rlink = newNode;
    return front;
}

NODE deleteF(NODE front) {
    if (front == NULL) {
        printf("\nList is Empty\n");
        return front;
    }
    if (front->Rlink == NULL) {
        free(front);
        return NULL;
    }
    NODE temp =front;
    front=temp->Rlink;
    free(temp);
    return front;
}

NODE deleteR(NODE front) {
    if (front == NULL) {
        printf("\nList is Empty\n");
        return front;
    }
    if (front->Rlink == NULL) {
        free(front);
        return NULL;
    }
    NODE temp=front;
```

```
    NODE prev=NULL;
    while (temp->Rlink != NULL) {
            prev=temp;
        temp = temp->Rlink;
    }
    prev->Rlink = temp->Rlink;
    free(temp);
    return front;
}

void display(NODE front) {
    if (front == NULL) {
        printf("\nList is Empty\n");
        return;
    }
    NODE temp = front;
    do {
        printf("%s \t", temp->BT);
        printf("%s \t", temp->AN);
        printf("%d \n", temp->Edi);
        temp = temp->Rlink;
    } while (temp !=NULL );
    printf("\n");
}
void countNode(NODE front)
{
    if (front == NULL) {
        printf("\nList is Empty\n");
        return;
    }
    int count=0;
    NODE temp=front;
    do
    {
        count++;
        temp=temp->Rlink;
    }while (temp!=NULL );
    printf("Number of Nodes=%d",count);
}
int main() {
    int ch;
```

```
    NODE front = NULL;
    for (;;) {
        printf("\n\tMenu\n");
        printf("1: To insert Front\n");
        printf("2: To insert Rear\n");
        printf("3: To Delete Front\n");
        printf("4: To Delete Rear\n");
        printf("5: To Display\n");
        printf("6: To count\n");
        printf("7: To EXIT\n");
        printf("Enter your Choice :\n");
        scanf("%d", &ch);
        switch (ch) {
            case 1:
                front = insertF(front, getNode());
                break;
            case 2:
                front = insertR(front, getNode());
                break;
            case 3:
                front= deleteF(front);
                break;
            case 4:
                front= deleteR(front);
                break;
            case 5:
                display(front);
                break;
            case 6:
                countNode(front);
                break;
            case 7:
                exit(0);
            default:
                printf("\nInvalid Choice\n");
        }
    }
    return 0;
}
```

Output:

```
        Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
1
Enter the Book Name:
Principles of c programming
Enter the author Name:
Reema Tareja
Enter the Book Edition:
2

        Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
2
Enter the Book Name:
Data Structures
Enter the author Name:
vasanth
Enter the Book Edition:
3

        Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
```

5: To Display
6: To count
7: To EXIT
Enter your Choice :
5
Principles of c programming Reema Tareja 2
Data Structures vasanth 3

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
2
Enter the Book Name:
Algorithms
Enter the author Name:
Padma Reddy
Enter the Book Edition:
5

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
1
Enter the Book Name:
Introduction to C++
Enter the author Name:
S K Uma
Enter the Book Edition:

1

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
5
Introduction to C++ S K Uma 1
Principles of c programming Reema Tareja 2
Data Structures vasanth 3
Algorithms Padma Reddy 5

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
4

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
5
Introduction to C++ S K Uma 1
Principles of c programming Reema Tareja 2

Data Structures vasanth 3

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
6
Number of Nodes=3
Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
3

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
5
Principles of c programming Reema Tareja 2
Data Structures vasanth 3

Menu
1: To insert Front

2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
7

Process returned 0 (0x0)   execution time : 251.653 s
Press any key to continue.

Program 4:

Implement a menu driven Program for the following operations on Circular Linked List.
i. Create CLL with information field of type string
ii. Count the number of nodes in the CLL.
iii. Delete the node at front/rear.
iv. Display the contents of CLL.

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

typedef struct Node {
   char data[100];
   struct Node *link;
} Node;

typedef Node* NODE;

NODE getNode() {
   NODE NewNode = (NODE)malloc(sizeof(Node));
   if (!NewNode) {
      printf("Memory Allocation Failed\n");
      exit(EXIT_FAILURE);
   }
   printf("Enter the integer Data:\n");
   scanf(" %99[^\n]", &NewNode->data);
   NewNode->link = NULL;
   return NewNode;
}

NODE insertF(NODE rear, NODE newNode) {
   if (rear == NULL) {
      newNode->link = newNode;
      return newNode;
   }
   newNode->link = rear->link;
   rear->link = newNode;
   return rear;
}

NODE insertR(NODE rear, NODE newNode) {
   if (rear == NULL) {
      newNode->link = newNode;
```

```
        return newNode;
    }
    newNode->link = rear->link;
    rear->link = newNode;
    return newNode;
}

NODE deleteR(NODE rear) {
    if (rear == NULL) {
        printf("\nList is Empty\n");
        return rear;
    }
    if (rear->link == rear) {
        free(rear);
        return NULL;
    }
    NODE temp = rear->link;
    while (temp->link != rear) {
        temp = temp->link;
    }
    temp->link = rear->link;
    free(rear);
    rear = temp;
    return rear;
}

NODE deleteF(NODE rear) {
    if (rear == NULL) {
        printf("\nList is Empty\n");
        return rear;
    }
    if (rear->link == rear) {
        free(rear);
        return NULL;
    }
    NODE front = rear->link;
    rear->link = front->link;
    free(front);
    return rear;
}
```

```
void display(NODE rear) {
    if (rear == NULL) {
        printf("\nList is Empty\n");
        return;
    }
    NODE temp = rear->link;
    do {
        printf("%s ", temp->data);
        temp = temp->link;
    } while (temp != rear->link);
    printf("\n");
}
void countNode(NODE rear)
{
    int count=0;
    NODE temp=rear->link;
    do
    {
        count++;
        temp=temp->link;
    }while(temp!=rear->link);
    printf("Number of Nodes=%d",count);
}
int main() {
    int ch;
    NODE rear = NULL;
    for (;;) {
        printf("\n\tMenu\n");
        printf("1: To insert Front\n");
        printf("2: To insert Rear\n");
        printf("3: To Delete Front\n");
        printf("4: To Delete Rear\n");
        printf("5: To Display\n");
        printf("6: To count\n");
        printf("7: To EXIT\n");
        printf("Enter your Choice :\n");
        scanf("%d", &ch);
        switch (ch) {
            case 1:
                rear = insertF(rear, getNode());
                break;
```

```
            case 2:
                rear = insertR(rear, getNode());
                break;
            case 3:
                rear = deleteF(rear);
                break;
            case 4:
                rear = deleteR(rear);
                break;
            case 5:
                display(rear);
                break;
            case 6:
                countNode(rear);
                break;
            case 7:
                exit(0);
            default:
                printf("\nInvalid Choice\n");
        }
    }
    return 0;
}
```

Output:

```
        Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
1
Enter the integer Data:
2

        Menu
1: To insert Front
2: To insert Rear
```

3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
1
Enter the integer Data:
1

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
2
Enter the integer Data:
3

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
5
1 2 3

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display

6: To count
7: To EXIT
Enter your Choice :
3

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
5
2 3

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
4

Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
5
2

Menu

```
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
6
Number of Nodes=1
        Menu
1: To insert Front
2: To insert Rear
3: To Delete Front
4: To Delete Rear
5: To Display
6: To count
7: To EXIT
Enter your Choice :
7

Process returned 0 (0x0)   execution time : 34.948 s
Press any key to continue.
```

Program 5:

Implement a menu driven Program for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)
i. Push an Element on to Stack (Handle the situation of overflow)
ii. Pop an Element from Stack (Handle the situation of underflow)
iii. Display the contents of Stack

```
#include <stdio.h>
#include <stdlib.h>
#define max 4

void push(int stack[],int *top)
{
   int item;
   if(*top==max-1)
   {
      printf("Stack is full");
      return;
   }
   printf("Enter element to insert:");
   scanf("%d",&item);
   stack[++(*top)]=item;
   printf("element=%d inserted successfully ",stack[*top]);
   return;
}
void pop(int stack[],int *top)
{
   int item;
   if(*top==-1 )
   {
      printf("Stack is empty");
      return;
   }

   item=stack[(*top)--];
   printf("element=%d deleted successfully ",item);
   return;
}
void Display(int stack[],int *top)
{
   if(*top==-1)
```

```
    {
    printf("Stack is empty");
    }
    else
    for(int i=0;i<=*top;i++)
    {
        printf("element-%d=%d  ",i,stack[i]);
    }

        return;
}

int main()
{
     int top=-1;
     int stack[max];
    for(;;)
    {
    int ch;

    printf("\n\tMenu\n");
    printf("1.Push\n");
    printf("2.POP\n");
    printf("3.Display\n");
    printf("Enter your Choice : ");
    scanf("%d",&ch);
    switch(ch)
    {
    case 1:
      push(stack,&top);
      break;
      case 2:
      pop(stack,&top);
      break;
      case 3:
       Display(stack,&top);
       break;
      case 4:
       exit(0);
       default:
       printf("\nInvalid choice\n");
```

```
   }
}
   return 0;
}
```

Output:

```
      Menu
1.Push
2.POP
3.Display
Enter your Choice : 1
Enter element to insert:1
element=1 inserted successfully
      Menu
1.Push
2.POP
3.Display
Enter your Choice : 1
Enter element to insert:2
element=2 inserted successfully
      Menu
1.Push
2.POP
3.Display
Enter your Choice : 1
Enter element to insert:3
element=3 inserted successfully
      Menu
1.Push
2.POP
3.Display
Enter your Choice : 3
element-0=1  element-1=2  element-2=3
      Menu
1.Push
2.POP
3.Display
Enter your Choice : 2
element=3 deleted successfully
      Menu
```

1.Push
2.POP
3.Display
Enter your Choice : 3
element-0=1 element-1=2
Menu
1.Push
2.POP
3.Display
Enter your Choice : 2
element=2 deleted successfully
Menu
1.Push
2.POP
3.Display
Enter your Choice : 3
element-0=1
Menu
1.Push
2.POP
3.Display
Enter your Choice : 2
element=1 deleted successfully
Menu
1.Push
2.POP
3.Display
Enter your Choice : 3
Stack is empty
Menu
1.Push
2.POP
3.Display
Enter your Choice : 2
Stack is empty
Menu
1.Push
2.POP
3.Display
Enter your Choice : 4

Program -6:

Implement a Program to convert an infix expression to its equivalent postfix expression.

```
#include <stdio.h>
#include <stdlib.h>
#define Max 50
int isOptr(char c){
   return(c=='+'||c=='-'||c=='*'||c=='/'||c=='^'||c=='$');
}
int isOpd(char c){
   return((c>='A' && c<='Z')||(c>='a' && c<='z')||(c>='0' && c<='9'));
}
int getPriority(char c)
{
   switch(c)
   {
   case '+':
   case '-':
      return 1;
   case '*':
   case '/':
      return 2;
   case '$':
   case '^':
      return 3;
   default:
      return 0;
   }
}
void I2P(char Infix[] , char Po[])
{
   int i,j,top=-1;
   char stack[Max];
   stack[++top]='#';
  for(j=0,i=0;Infix[i]!='\0';i++)
  {
   char sym=Infix[i];
   if(isOpd(sym))
   {
    Po[j++]=sym;
```

```
    }
    else if(sym=='(')
    {
      stack[++top]=sym;
    }
    else if(sym==')')
    {
        while(top!=-1 && stack[top]!='(' )
        {
           Po[j++]=stack[top--];
        }
        top--;
    }
     else if(isOptr(sym))
     {
         while(top!=-1 && getPriority(stack[top])>=getPriority(sym)&&stack[top]!='#' )
         {
            Po[j++]=stack[top--];
         }
         stack[++top]=sym;
     }
  }
  while(stack[top]!='#')
  {
     Po[j++]=stack[top--];
  }
  Po[j]='\0';
}
int main()
{
    char Infix[100],Po[100];
    printf("Enter Infix Expression\n");
    scanf(" %99[^\n]",Infix);
    I2P(Infix,Po);
    puts(Po);
    return 0;
}
```

Output:

```
Enter Infix Expression
1+(2*3)/3
123*3/+
```

Program 7:

Implement the following using recursion:
Tower_of_Hanoi
GCD of two numbers
Largest of 'n' numbers

```
#include <stdio.h>
#include <stdlib.h>
int GCD(int m ,int n)
{
   if(n==0)
      return m;
   GCD(n,m%n);
}
void Hanoi(int n,char s, char a , char d )
{
   if(n==1)
   {
      printf("\nmove disk 1 from %c to %c",s,d);
            return;
   }
   Hanoi(n-1,s,d,a);
   printf("\nmove disk %d from %c to %c",n,s,d);
   Hanoi(n-1,a,s,d);
}

int Large(int a[] , int n)
{
   if(n==1)
      return a[0];
   int max=Large(a+1,n-1);
   return (a[0]>max?a[0]:max);
}
int main() {
   int ch,a,b,n,num,res1,res2,ar[100];
   for (;;) {
      printf("\n\tMenu\n");
      printf("1: To GCD\n");
      printf("2: To Tower of hanoi\n");
      printf("3: To Largest\n");
      printf("Enter your Choice :\n");
      scanf("%d", &ch);
```

```
        switch (ch) {
            case 1:
                printf("Enter the value of a and b :");
                scanf("%d%d",&a,&b);
                 res1=GCD(a,b);
                printf("GCD=%d",res1);
                break;
            case 2:
                printf("Enter the number of disks:");
                scanf("%d",&num);
               Hanoi(num,'A','B','C');
                break;
            case 3:
                printf("Enter the size of array:");
                scanf("%d",&n);
                 printf("Enter the elements:");
                 for(int i=0;i<n;i++)
                scanf("%d",&ar[i]);
                 res2=Large(ar,n);
                printf("Largest=%d",res2);
                break;
            case 4:
                exit(0);
            default:
                printf("\nInvalid Choice\n");
        }
    }
    return 0;
}
```

Output:

```
        Menu
1: To GCD
2: To Tower of hanoi
3: To Largest
Enter your Choice :     1
Enter the value of a and b :     6          12
GCD=6
        Menu
1: To GCD
2: To Tower of hanoi
3: To Largest
Enter your Choice :
2
Enter the number of disks:3

move disk 1 from A to C
move disk 2 from A to B
move disk 1 from C to B
move disk 3 from A to C
move disk 1 from B to A
move disk 2 from B to C
move disk 1 from A to C
        Menu
1: To GCD
2: To Tower of hanoi
3: To Largest
Enter your Choice :
3
Enter the size of array:4
Enter the elements:90
23
-76
5
Largest=90
        Menu
1: To GCD
2: To Tower of hanoi
3: To Largest
Enter your Choice :
4
```

Program 8:

Implement a menu driven Program for the following operations on QUEUES of Strings using Linked list
i. Insert an Element into Queue
ii. Delete an Element from Queue
iii. Display the contents of Queue

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node
{
   char data[100];
   struct Node *link;
}*NODE;

NODE front=NULL , rear=NULL;
NODE getNode()
{
NODE    NewNode=(NODE)malloc(sizeof(struct Node));
   if(!NewNode)
      printf("Memory Allocation Failed\n");
   printf("Enter the integer Data:\n");
   scanf(" %99[^\n]",&NewNode->data);
   NewNode->link=NULL;
   return NewNode;
}
NODE insert(NODE newNode)
{
   if(front==NULL)
   {
      front=rear=newNode;
   }
   rear->link=newNode;
   rear=newNode;
}
NODE Delete()
{
    if(front==NULL)
    {
       printf("List is empty\n");
    }
   NODE temp=front;
```

```
    NODE prev=NULL;
    if(temp->link==NULL)
    {
        free(temp);
        front=rear=NULL;
    }
    else{
        front=temp->link;
        free(temp);
    }
}

void Display()
{
    if(front==NULL)
     {
        printf("\nList is Empty");
     }
    else
        {
    NODE temp=front;
   do
   {
      printf(" %s",temp->data);
      temp=temp->link;
   }while(temp!=rear->link);

}
}
int main()
{
    int ch;

    for(;;)
    {

    printf("\n\tMenu\n");
    printf("1: To insert Front\n");
    printf("2: To Delete Element\n");
    printf("3: To Display\n");
    printf("Enter your Choice :\n");
```

```
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
    insert(getNode());
    break;
  case 2:
    Delete();
    break;
    case 3:
    Display();
    break;
    case 4:
        exit(0);
    default:
        printf("\nInvalid Choice\n");

        }
    }

    return 0;
}
```

Output:

```
        Menu
1: To insert Front
2: To Delete Element
3: To Display
Enter your Choice :
1
Enter the integer Data:
2

        Menu
1: To insert Front
2: To Delete Element
3: To Display
Enter your Choice :
1
Enter the integer Data:
```

3

Menu
1: To insert Front
2: To Delete Element
3: To Display
Enter your Choice :
3
2 3
Menu
1: To insert Front
2: To Delete Element
3: To Display
Enter your Choice :
2

Menu
1: To insert Front
2: To Delete Element
3: To Display
Enter your Choice :
3
3
Menu
1: To insert Front
2: To Delete Element
3: To Display
Enter your Choice :
4

9. Implement a menu driven program to perform the following operations on priority queue using linked list.
i. Insert a node based on priority.
ii. Delete a node from the queue
iii. Display the contents of the queue

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

typedef struct Node {
   int data;
   int pri;
   struct Node *link;
} Node;

typedef Node* NODE;

NODE front = NULL, rear = NULL;

NODE getNode() {
   NODE NewNode = (NODE)malloc(sizeof(Node));
   if (!NewNode) {
      printf("Memory Allocation Failed\n");
      exit(1);
   }
   printf("Enter the integer Data:\n");
   scanf("%d", &NewNode->data);
   printf("Enter the Priority of Data:\n");
   scanf("%d", &NewNode->pri);
   NewNode->link = NULL;
   return NewNode;
}

void insert(NODE newNode) {
   if (front == NULL) {
      front = rear = newNode;
   } else {
      NODE first = front;
      NODE last = rear;
      NODE prev = NULL;
      if (newNode->pri < first->pri) {
         newNode->link = first;
         front = newNode;
```

```
        } else if (newNode->pri >= last->pri) {
            last->link = newNode;
            rear = newNode;
        } else {
            while (newNode->pri >= first->pri) {
                prev = first;
                first = first->link;
            }
            prev->link = newNode;
            newNode->link = first;
        }
    }
}

void Delete() {
    if (front == NULL) {
        printf("List is empty\n");
        return;
    }
    NODE temp = front;
    if (temp->link == NULL) {
        free(temp);
        front = rear = NULL;
    } else {
        front = temp->link;
        free(temp);
    }
}

void Display() {
    if (front == NULL) {
        printf("\nList is Empty\n");
    } else {
        NODE temp = front;
        while (temp != NULL) {
            printf("\nData=%d\tPriority=%d", temp->data, temp->pri);
            temp = temp->link;
        }
    }
}
```

```
int main() {
   int ch;

   for (;;) {
      printf("\n\tMenu\n");
      printf("1: To insert \n");
      printf("2: To Delete Element\n");
      printf("3: To Display\n");
      printf("4: To Exit\n");
      printf("Enter your Choice :\n");
      scanf("%d", &ch);
      switch (ch) {
         case 1:
            insert(getNode());
            break;
         case 2:
            Delete();
            break;
         case 3:
            Display();
            break;
         case 4:
            exit(0);
         default:
            printf("\nInvalid Choice\n");
      }
   }
   return 0;
}
```

Output:

```
      Menu
1: To insert
2: To Delete Element
3: To Display
4: To Exit
Enter your Choice :
1
Enter the integer Data:
1
Enter the Priority of Data:
```

```
3

        Menu
1: To insert
2: To Delete Element
3: To Display
4: To Exit
Enter your Choice :
1
Enter the integer Data:
0
Enter the Priority of Data:
3

        Menu
1: To insert
2: To Delete Element
3: To Display
4: To Exit
Enter your Choice :
1
Enter the integer Data:
6
Enter the Priority of Data:
8

        Menu
1: To insert
2: To Delete Element
3: To Display
4: To Exit
Enter your Choice :
3

Data=1  Priority=3
Data=0  Priority=3
Data=6  Priority=8
        Menu
1: To insert
2: To Delete Element
3: To Display
```

4: To Exit
Enter your Choice :
2

Menu
1: To insert
2: To Delete Element
3: To Display
4: To Exit
Enter your Choice :
3

Data=0 Priority=3
Data=6 Priority=8
Menu
1: To insert
2: To Delete Element
3: To Display
4: To Exit
Enter your Choice :
2

Menu
1: To insert
2: To Delete Element
3: To Display
4: To Exit
Enter your Choice :
3

Data=6 Priority=8
Menu
1: To insert
2: To Delete Element
3: To Display
4: To Exit
Enter your Choice :
4

Program :10

Implement a menu driven Program to perform the following operations on Binary Search Tree (BST)
i. Create a BST of N Integers
ii.Tree Traversals methods

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
   int data;
   struct Node *Llink;
   struct Node *Rlink;
} Node;
typedef Node* NODE;

NODE front = NULL, rear = NULL;
NODE getNode() {
   NODE NewNode = (NODE)malloc(sizeof(Node));
   if (!NewNode) {
      printf("Memory Allocation Failed\n");
      exit(1);
   }
   printf("Enter the integer Data:\n");
   scanf("%d", &NewNode->data);
   NewNode->Llink=NewNode->Rlink = NULL;
   return NewNode;
}

NODE insert(NODE Root,NODE newNode) {
   if (Root == NULL) {
      return newNode;
   }
   NODE temp=Root;
      if (newNode->data >temp->data )
            temp->Rlink=insert(temp->Rlink,newNode);
      else
         temp->Llink=insert(temp->Llink,newNode);
         return Root;
}

void PreOrder(NODE Root) {
      if(Root!=NULL)
      {
```

```
            printf("%d ",Root->data);
            PreOrder(Root->Llink);
            PreOrder(Root->Rlink);
        }

}
void PostOrder(NODE Root) {

        if(Root!=NULL)
        {

            PostOrder(Root->Llink);
            PostOrder(Root->Rlink);
            printf("%d ",Root->data);
        }
}

void InOrder(NODE Root) {
    if(Root!=NULL)
        {

            InOrder(Root->Llink);
            printf("%d ",Root->data);
            InOrder(Root->Rlink);
        }
}
int main() {
    int ch;
NODE Root=NULL;
    for (;;) {
        printf("\n\tMenu\n");
        printf("1: To insert Node\n");
        printf("2: To PreOrder Traversal\n");
        printf("3: To PostOrder Traversal\n");
        printf("4: To To InOrder Traversal\n");
        printf("5: To Exit\n");
        printf("Enter your Choice :\n");
        scanf("%d", &ch);
        switch (ch) {
            case 1:
                Root=insert(Root,getNode());
```

```
            break;
        case 2:
        PreOrder(Root);
            break;
        case 3:
            PostOrder(Root);
            break;
             case 4:
            InOrder(Root);
            break;
        case 5:
            exit(0);
        default:
            printf("\nInvalid Choice\n");
      }
   }
   return 0;
}
```

Output:

```
      Menu
1: To insert Node
2: To PreOrder Traversal
3: To PostOrder Traversal
4: To To InOrder Traversal
5: To Exit
Enter your Choice :
1
Enter the integer Data:
5

      Menu
1: To insert Node
2: To PreOrder Traversal
3: To PostOrder Traversal
4: To To InOrder Traversal
5: To Exit
Enter your Choice :
1
Enter the integer Data:
4
```

Menu
1: To insert Node
2: To PreOrder Traversal
3: To PostOrder Traversal
4: To To InOrder Traversal
5: To Exit
Enter your Choice :
1
Enter the integer Data:
6

Menu
1: To insert Node
2: To PreOrder Traversal
3: To PostOrder Traversal
4: To To InOrder Traversal
5: To Exit
Enter your Choice :
1
Enter the integer Data:
2

Menu
1: To insert Node
2: To PreOrder Traversal
3: To PostOrder Traversal
4: To To InOrder Traversal
5: To Exit
Enter your Choice :
1
Enter the integer Data:
1

Menu
1: To insert Node
2: To PreOrder Traversal
3: To PostOrder Traversal
4: To To InOrder Traversal
5: To Exit
Enter your Choice :

1
Enter the integer Data:
7

Menu
1: To insert Node
2: To PreOrder Traversal
3: To PostOrder Traversal
4: To To InOrder Traversal
5: To Exit
Enter your Choice :
1
Enter the integer Data:
8

Menu
1: To insert Node
2: To PreOrder Traversal
3: To PostOrder Traversal
4: To To InOrder Traversal
5: To Exit
Enter your Choice :
2
5 4 2 1 6 7 8
Menu
1: To insert Node
2: To PreOrder Traversal
3: To PostOrder Traversal
4: To To InOrder Traversal
5: To Exit
Enter your Choice :
3
1 2 4 8 7 6 5
Menu
1: To insert Node
2: To PreOrder Traversal
3: To PostOrder Traversal
4: To To InOrder Traversal
5: To Exit
Enter your Choice :
4

1 2 4 5 6 7 8

Menu

1: To insert Node

2: To PreOrder Traversal

3: To PostOrder Traversal

4: To To InOrder Traversal

5: To Exit

Enter your Choice :

5

##########ALL THE BEST##########