# INSERTION SORT

**PROGRAM CODE:-**

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
void main()
{
 int i,j,n,temp,a[10];
 printf("enter the number of elements");
 scanf("%d",&n);
 printf("elements in array is:");
 for(i=0;i<n;i++)
 {scanf("%d",&a[i]);
 }
 for(i=0;i<n;i++)
  {
    temp=a[i];
    j=i-1;
     while(j>=0&&a[j]>temp)
      {
       a[j+1]=a[j];
       j--;

      }
     a[j+1]=temp;
  }
printf("sorted elements are:");
for(i=0;i<n;i++)
 { printf("%d\t",a[i]);
 }


}
```

**OUTPUT**

```
enter the number of elements 6
elements in array is:4 3 2 6 7 8
sorted elements are:2      3      4      6      7      8
```

# BINARY SEARCH

**PROGRAM CODE:-**

```
#include<stdio.h>
int main()
{int i,j,temp,n;
 int left,right,mid;
 int flag=0,key,a[10];
 printf("enter the number of elements\n");
 scanf("%d",&n);
 left=0;right=n-1;mid=0;
 printf("elements in the array \n");
 for(i=0;i<n;i++)
 {scanf("%d",&a[i]);
 }
 for(i=0;i<n-1;i++)
  {
    for(j=0;j<n-i-1;j++)
     {if(a[j]>a[j+1])
      { temp=a[j];
      a[j]=a[j+1];
      a[j+1]=temp;
      }
     }
 }
 printf("enter the element which is needed to find\n");
 scanf("%d",&key);
 while(left<=right)h
  {
   mid=(left+right)/2;
   if(a[mid]==key)
    {flag=1;
     break;}
    else if(a[mid]<key)
     {left=mid+1;
      }
    else
     {right=mid-1;
     }
    }

 if(flag==1)
  {
    printf("the element is found at a position %d",mid+1);
   }
```

```
 else
  {printf("element is not found");
   }
return 0;
}
```

**OUTPUT**

```
enter the number of elements
7
elements in the array
1 2 3 4 5 6 7
enter the element which is needed to find
5
the element is found at a position 5
```

# POLYNOMIAL ADDITION

## PROGRAM CODE:-

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
struct poly
{int coef;
 int exp;
}p[100];
void main()
{
 int i,j,k,m,n;
 printf("the number of elements in first polynomial:");
 scanf("%d",&m);
 printf("exponentials and coeficiant in ascending order is\n");
 for(i=0;i<m;i++)
  { printf("coefficiant\texponential\n");
    scanf("%d",&p[i].coef);

    scanf("%d",&p[i].exp);
  }
   printf("the number of elements in second polynomial:");
 scanf("%d",&n);
 printf("exponentials and coefficient in ascending order is\n");
 for(i=m;i<m+n;i++)
  { printf("coefficiant\texponential\n");
    scanf("%d",&p[i].coef);

    scanf("%d",&p[i].exp);

  }
 i=0;
 k=m+n;
 j=m;
 while(i<m&&j<m+n)
   {
    if(p[i].exp<p[j].exp)
      {
       p[k].exp=p[i].exp;
       p[k].coef=p[i].coef;
       k++;
       i++;
      }
```

```
   else if(p[j].exp<p[i].exp)
     {
       p[k].exp=p[j].exp;
       p[k].coef=p[j].coef;
       k++;
       j++;

     }
   else
   {
     p[k].exp=p[j].exp;
     p[k].coef=p[i].coef+p[j].coef;
     k++;
     i++;
     j++;
   }

 }
 while(i<m)

    {
     p[k].exp=p[i].exp;
     p[k].coef=p[i].coef;
     k++;
     i++;
    }

 while(j<m+n)

  { p[k].exp=p[j].exp;
       p[k].coef=p[j].coef;
       k++;
       j++;
 }


 printf("addition of polynomial is:");
 for(i=k-1;i>=m+n;i--)
  {
   printf("%dX^(%d)+",p[i].coef,p[i].exp);
  }
}
```

**OUTPUT**

```
the number of elements in first polynomial:4
exponentials and coeficiant in ascending order is
coefficiant exponential
3               2
coefficiant exponential
1               4
coefficiant exponential
5               5
coefficiant exponential
3               6
the number of elements in second polynomial:3
exponentials and coefficient in ascending order is
coefficiant exponential
4               1
coefficiant exponential
3               4
coefficiant exponential
5               5
addition of polynomial is:3X^(6)+10X^(5)+4X^(4)+3X^(2)+4X^(1)
```

# SPARSE MATRIX

**PROGRAM CODE;-**

```
//SANIN MOHAMMED N
//B21CSB55

#include <stdio.h>

struct sparse

{   int row,col;

    int a[10][10];

    int tuple[100][3];

};

void readarray(struct sparse *sp,int i)

{printf("Enter no of rows and columns of matrix %d\n",i);

scanf("%d%d",&sp->row,&sp->col);

printf("Enter the elements\n");

for (int i=0;i<sp->row;i++)

for (int j=0;j<sp->col;j++)

scanf("%d",&sp->a[i][j]);

}
void disparray(struct sparse *sp,int i)

{printf("The matrix %d\n",i);

  for (int i=0;i<sp->row;i++)

 { for (int j=0;j<sp->col;j++)

    {printf("%d  ",sp->a[i][j]);}

  printf("\n");
```

```
  }
}
void disptuple(struct sparse *sp,int i)
{printf("The tuple representation of sparse matrix %d\n",i);
  for (int i=0;i<=sp->tuple[0][2];i++)
  { for (int j=0;j<3;j++)
    {printf("%d  ",sp->tuple[i][j]);}
  printf("\n");
  }
}
void maketuple(struct sparse *sp)
{int k=0;
sp->tuple[0][0]=sp->row;
sp->tuple[0][1]=sp->col;
for (int i=0;i<sp->row;i++)
for (int j=0;j<sp->col;j++)
{if(sp->a[i][j]!=0)
  {k++;
   sp->tuple[k][0]=i;
   sp->tuple[k][1]=j;
   sp->tuple[k][2]=sp->a[i][j];
  }
  sp->tuple[0][2]=k;
```

```
}
}
void transtuple(struct sparse *sp1,struct sparse *sp2,int a)
{
if (sp1->tuple[0][2]==0)
printf("Matrix %d cannot be transposed",a);
else
{
sp2->tuple[0][1]=sp1->tuple[0][0];
sp2->tuple[0][0]=sp1->tuple[0][1];
sp2->tuple[0][2]=sp1->tuple[0][2];
int k=1;
for(int i=0;i<sp1->tuple[0][1];i++)
for(int j=1;j<=sp1->tuple[0][2];j++)
if(i==sp1->tuple[j][1])
{
sp2->tuple[k][0]=sp1->tuple[j][1];
sp2->tuple[k][1]=sp1->tuple[j][0];
sp2->tuple[k][2]=sp1->tuple[j][2];
k++;
}
printf("Transpose is\n");
```

```
disptuple(sp2,a);

}}

void addtuple(struct sparse *sp1,struct sparse *sp2,struct sparse *sp3,int a,int b)

{

int i=1,j=1,k=1;

if(sp1->tuple[0][0]!=sp2->tuple[0][0]||sp1->tuple[0][1]!=sp2->tuple[0][1])

printf("Matrix %d and Matrix %d cannot be added\n",a,b);

else
{
while (i<=sp1->tuple[0][2]||j<=sp2->tuple[0][2])

{

if(i>sp1->tuple[0][2])
{

sp3->tuple[k][0]=sp2->tuple[j][0];

sp3->tuple[k][1]=sp2->tuple[j][1];

sp3->tuple[k][2]=sp2->tuple[j][2];

k++;j++;

}

else if(j>sp2->tuple[0][2])
{
sp3->tuple[k][0]=sp1->tuple[i][0];

sp3->tuple[k][1]=sp1->tuple[i][1];

sp3->tuple[k][2]=sp1->tuple[i][2];

k++;i++;

}
else if(sp1->tuple[i][0]==sp2->tuple[j][0])
```

```
{
if(sp1->tuple[i][1]==sp2->tuple[j][1])
{
sp3->tuple[k][0]=sp2->tuple[j][0];
sp3->tuple[k][1]=sp2->tuple[j][1];
sp3->tuple[k][2]=sp2->tuple[j][2]+sp1->tuple[i][2];
k++;i++;j++;
}
else if(sp1->tuple[i][1]<sp2->tuple[j][1])
{
sp3->tuple[k][0]=sp1->tuple[i][0];
sp3->tuple[k][1]=sp1->tuple[i][1];
sp3->tuple[k][2]=sp1->tuple[i][2];
k++;i++;
}
else
{
sp3->tuple[k][0]=sp2->tuple[j][0];
sp3->tuple[k][1]=sp2->tuple[j][1];
sp3->tuple[k][2]=sp2->tuple[j][2];
k++;j++;
}
```

```
}
else if(sp1->tuple[i][0]<sp2->tuple[j][0])
{
sp3->tuple[k][0]=sp1->tuple[i][0];
sp3->tuple[k][1]=sp1->tuple[i][1];
sp3->tuple[k][2]=sp1->tuple[i][2];
k++;i++;
}
else
{
sp3->tuple[k][0]=sp2->tuple[j][0];
sp3->tuple[k][1]=sp2->tuple[j][1];
sp3->tuple[k][2]=sp2->tuple[j][2];
k++;j++;
}
}
sp3->tuple[0][0]=sp1->tuple[0][0];
sp3->tuple[0][1]=sp1->tuple[0][1];
sp3->tuple[0][2]=k-1;
printf("Sum of Matrices %d and %d\n",a,b);
disptuple(sp3,3);
}
}
```

```
void main()

{

struct sparse sp1,sp2,trans1,trans2,sumsp3;
readarray(&sp1,1);
readarray(&sp2,2);
maketuple(&sp1);
maketuple(&sp2);
disptuple(&sp1,1);

disptuple(&sp2,2);
transtuple(&sp1,&trans1,1);
transtuple(&sp2,&trans2,2);
addtuple(&sp1,&sp2,&sumsp3,1,2);

}
```

**OUTPUT**

```
Enter no of rows and columns of matrix 1
3 2
Enter the elements
0 2
3 4
0 9
Enter no of rows and columns of matrix 2
3 2
Enter the elements
1 2
0 4
5 0
The tuple representation of sparse matrix 1
3  2  4
0  1  2
1  0  3
1  1  4
2  1  9
The tuple representation of sparse matrix 2
3  2  4
0  0  1
0  1  2
1  1  4
2  0  5


Transpose is
```

The tuple representation of sparse matrix 1
2 3 4
0 1 3
1 0 2
1 1 4
1 2 9
Transpose is
The tuple representation of sparse matrix 2
2 3 4
0 0 1
0 2 5
1 0 2
1 1 4
Sum of Matrices 1 and 2
The tuple representation of sparse matrix 3
3 2 6
0 0 1
0 1 4
1 0 3
1 1 8
2 0 5
2 1 9

# STACK

**PROGRAM CODE:-**

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
#include<stdlib.h>
int N,top=-1,stack[100];

void push(int X)
{
 if(top==(N-1))
   {
    printf("stack is full \n");

   }
 else
  {
    top++;
    stack[top]=X;
   }
}

int pop()
{
 if(top==-1)
  {printf("stack is empty");
   }
 else
  {    top--;
    return stack[top+1];


   }
}

void display()
{
  if(top==-1)
    {
      printf("stack is empty");
     }
  else
   { printf("the stacK is:");
     for(int i=0;i<=top;i++)
      {
```

```
        printf("%d\t",stack[i]);
       }
    }
}

void main()
 {
   int choice,X;
   printf("the number of elements in stack:");
   scanf("%d",&N);

   while(1)
   {
    printf("\n1:push\n2:pop\n3:display\n4:exit \nother:invalid \n select from 1-4\n");
    scanf("%d",&choice);

    switch(choice)
     {
      case 1:
       { printf("enter the element to add:");
         scanf("%d",&X);
         push(X);
         display();
         break;
        }
      case 2:
      { printf("the removed element is %d",pop());

        display();
        break;
      }
      case 3:
      {display();
       break;
      }
      case 4:
       {exit(0);
        break;
       }
      default:
      {printf("invalid choice");
       break;
      }}}}
```

**OUTPUT**

```
the number of elements in stack:4
```

```
1:push
2:pop
3:display
4:exit
other:invalid
 select from 1-4
1
enter the element to add:5
the stacK is:5
1:push
2:pop
3:display
4:exit
other:invalid
 select from 1-4
1
enter the element to add:4
the stacK is:5  4
1:push
2:pop
3:display
4:exit
other:invalid
 select from 1-4
1
enter the element to add:3
the stacK is:5  4    3
1:push
2:pop
3:display
4:exit
other:invalid
 select from 1-4
1
enter the element to add:9
the stacK is:5  4    3    9
1:push
2:pop
3:display
4:exit
other:invalid
 select from 1-4
2
the removed element is 9
the stacK is:5  4    3
```

1:push
2:pop
3:display
4:exit
other:invalid
 select from 1-4
2
the removed element is 3
the stacK is:5  4
1:push
2:pop
3:display
4:exit
other:invalid
 select from 1-4
4

# INFIX TO POSTFIX BY USING STACK

**PROGRAM CODE:-**

```
//SANIN MOHAMMED N
//B21CSB55


#include <stdio.h>

#include <string.h>

#include <math.h>

#include <ctype.h>


int priority(char ch)

{switch(ch)

{case '+':return 1;break;

case '-':return 1;break;

case '*':return 2;break;

case '/':return 2;break;

case '^':return 3;break;

case '(':return 0;break;

}return 0;

}


void main()

{int len,i=0,o=0,top=-1,j,item1,item2,p1,p2,k=1,a[20];

char stack[100],output[100],ch[100];
```

```
printf("Give the input expression\n");
scanf("%s",ch);

 while (ch[i]!='\0')
{  if (isalpha(ch[i]))
    {output[o]=ch[i];
     i++;o++;
    }

    else if(ch[i]=='(')
    {top++;
     stack[top]=ch[i];
     i++;
    }

    else if(ch[i]==')')
    {   while(top!=-1)
       {   if (stack[top]=='(')
              break;
          output[o]=stack[top];
          o++;top--;
       }
        if (stack[top]=='(')
```

```
        top--;
      i++;
}

else
{   if(top==-1||stack[top]=='(')
    {top++;
     stack[top]=ch[i];
     i++;
    }

    else
    {p1=priority(stack[top]);
     p2=priority(ch[i]);

     if(p1<p2)
     {top++;
     stack[top]=ch[i];
     i++;
     }

     else if(p1==p2)
     {  if(ch[i]=='^')
```

```
   {top++;
   stack[top]=ch[i];
   i++;
   }

   else
   {output[o]=stack[top];
    stack[top]=ch[i];
    i++;o++;
   }
}

else
{  while(p1>=p2)
   {output[o]=stack[top];
    top--;o++;
    if(top==-1)
       break;
    p1=priority(stack[top]);
   }
top++;
stack[top]=ch[i];
```

```
        i++;
        }
       }
    }
}


 while(top!=-1)
{  if(stack[top]=='('||stack[top]==')')
          continue;
    else
    {output[o]=stack[top];
     top--;
     o++;
    }
}

 for(j=0;j<o;j++)
    {printf("%c",output[j]);
    }

 top=-1;
for(i=0;i<strlen(output);i++)
{  if(isalpha(output[i]))
```

```
{top++;
 printf("\nEnter the value of %c",output[i]);
 scanf("\n%d",&a[top]);
}

else if(output[i]=='+'||output[i]=='-'||output[i]=='*'||output[i]=='/'||output[i]=='^')
{item1=a[top];
 top--;
 item2=a[top];

 switch(output[i])
 {  case '+':a[top]=item2+item1;break;
    case '-':a[top]=item2-item1;break;
    case '*':a[top]=item2*item1;break;
    case '/':a[top]=item2/item1;break;
    case '^':
            for(j=0;j<item1;j++)
            {k*=item2;}
            a[top]=k;
            break;
} }}
printf("\nPostfix evaluation result is  :%d",a[0]);}
```

**OUTPUT**

Give the input expression
(a*b-c)/d
ab*c-d/
Enter the value of a 3
3

Enter the value of b 2
2

Enter the value of c 1
1

Enter the value of d 5
5

Postfix evaluation result is  :1

# PREFIX

## PROGRAM CODE

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
#include<ctype.h>
#include<string.h>

int ICP(char infix)
{
  if(infix=='+'||infix=='-')
  {
    return 2;
  }
  if(infix=='*'||infix=='/')
  {
    return 4;
  }
  if(infix=='^')
  {
    return 5;
  }


}

int ISP(char stack)
{
if(stack=='+'||stack=='-')
  {
    return 1;
  }
  if(stack=='*'||stack=='/')
  {
    return 3;
  }
  if(stack=='^')
  {
    return 6;
  }
  if(stack=='(')
  {
   return 0;
  }
```

```
}

void main()
{
  char infix[100],output[100],stack[100],reverse[100];
  int i=0,o=0,top=-1,j=0;
  int p1,p2;
  printf("enter the infix expression:");
  scanf("%s",infix);

  for(i=0;i<strlen(infix);i++)
  {
    if(infix[strlen(infix)-i-1]=='(')
    {
      reverse[i]=')';
    }
    else if(infix[strlen(infix)-i-1]==')')
    {
      reverse[i]='(';
    }
    else
    reverse[i]=infix[strlen(infix)-i-1];

  }
   i=0;
  while(reverse[i]!='\0')
  {
    if(isalpha(reverse[i]))
     {
       output[o]=reverse[i];
       i++;
       o++;
     }
     else if(reverse[i]==')')
     {
         while(top!=-1)

        {   if (stack[top]=='(')

                break;

            output[o]=stack[top];

            o++;top--;
```

```
        }

        if (stack[top]=='(')

           top--;

        i++;

    }

    else if(reverse[i]=='(')
    {
       top++;
     stack[top]=reverse[i];
     i++;

    }
    else if(reverse[i]=='+'||reverse[i]=='-
'||reverse[i]=='*'||reverse[i]=='/'||reverse[i]=='^')
    {
      p1=ICP(reverse[i]);
      p2=ISP(stack[top]);

      if(p1>p2)
      {
        top++;
        stack[top]=reverse[i];
        i++;

      }
      else
      {
       output[o]=stack[top];
       o++;
       stack[top]=reverse[i];
       i++;
      }

    }

  }

  while(top!=-1)
  { if(stack[top]=='(')
    {
```

```
      continue;
    }
    output[o]=stack[top];
    top--;
    o++;

  }
  printf("prefix expression is:");

  for(j=0;j<o;j++)
  {
    printf("%c",output[o-j-1]);

  }

}
```

**OUTPUT**

```
enter the infix expression:(a*b-c/d)
prefix expression is:-*ab/cd
```

# QUEUE

## PROGRAM CODE

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
#include<stdlib.h>

int N,front=-1,rear=-1,queue[100];

void enqueue(int X)
 {
  if(rear==(N-1))
   {
    printf("queue is full");
   }

  else if(front==-1&&rear==-1)
  { front=0;
    rear=0;
    queue[0]=X;
  }
 else
   {rear++;
    queue[rear]=X;
   }
 }
int dequeue()
{
 if(front==-1&&rear==-1)
  {
    printf("queue is empty");
   }
 else if(front==rear)
  {
    return queue[front];
    front=-1;
    rear=-1;
  }
 else
  {
   front++;
   return queue[front-1];
  }
```

```
}
void display()
 {
  if(front==-1&&rear==-1)
    {
      printf("queue is empty");
    }
  else
   { printf("the elements in queue is");
     for(int i=front;i<=rear;i++)
        {printf("%d\t",queue[i]);
         }
   }
 }


 void main()
 {
  int choice,X;
  printf("the number of elements in queue:");
  scanf("%d",&N);

  while(1)
  {
    printf("\n1:enqueue\n2:dequeue\n3:display\n4:exit \nother:invalid \n select from
1-4\n");
    scanf("%d",&choice);

    switch(choice)
     {
      case 1:
       { printf("enter the element to add:");
         scanf("%d",&X);
         enqueue(X);
         display();
         break;
        }
      case 2:
      {
        printf("the removed element is %d",dequeue());
        display();
        break;
      }
      case 3:
      {display();
```

```
        break;
       }
       case 4:
        {exit(0);
        }
       default:
       {printf("invalid choice");
       }
      }
   }
 }
```

**OUTPUT**

```
the number of elements in queue:5
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
 select from 1-4
1
enter the element to add:7
the elements in queue is7
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
 select from 1-4
1
enter the element to add:4
the elements in queue is7       4
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
 select from 1-4
1
enter the element to add:3
the elements in queue is  7      4         3
1:enqueue
2:dequeue
3:display
4:exit
```

other:invalid
select from 1-4
1
enter the element to add:8
the elements in queue is 7 4 3 8
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
select from 1-4
2
the removed element is 7
the elements in queue is 4 3 8
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
select from 1-4
2
the removed element is 4
the elements in queue is 3 8
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
select from 1-4
2
the removed element is 3
the elements in queue is 8
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
select from 1-4
4

# CIRCULAR QUEUE

## PROGRAM CODE

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
#include<stdlib.h>

int N;
int front=-1,rear=-1;
int queue[100];

void enqueue(int X)
{
 if((rear+1)%N==front)
 {
   printf("the queue is full!");
 }
else if(front==-1&&rear==-1)
 {
   front=0;
   rear=0;
   queue[0]=X;
 }
 else
{ rear=(rear+1)%N;
  queue[rear]=X;
 }
}

int dequeue()
{
 if(front==-1&&rear==-1)
 {
  printf("queue is empty!");
 }
 else if(front==rear)
 {
  front=-1;
  rear=-1;
  return queue[front+1];

 }
 else
{
  front=(front+1)%N;
```

```
  return queue[front-1];
}
}

void display()
{
  if(front==-1&&rear==-1)
   {
     printf("the queue is empty!");
   }
   else
   {printf("the queue is=");
    int i=front;
    while(i!=rear)
    {
     printf("%d(%d)\t",queue[i],i);
     i=(i+1)%N;
    }
   printf("%d(%d)\t",queue[rear],i);

   }
 }
 void main()
{
  int choice,X;
  printf("enter the no.of elements in the queue= ");
  scanf("%d",&N);

  while(1)
  {
     printf("\n1:enqueue\n2:dequeue\n3:display\n4:exit \nother:invalid \n select from
1-4\n");
     scanf("%d",&choice);
    switch(choice)
    {
      case 1:
       {
         printf("enter the element to be added:");
         scanf("%d",&X);
         enqueue(X);
         display();
         break;

       }
      case 2:
```

```
      {
        printf("the element deleted is:%d\n",dequeue());
        display();
         break;
      }
      case 3:
      {
       display();
       break;
      }
     case 4:
     {exit(0);
     }
     default:
     {
      printf("invalid choice");
     }
    }
  }
}
```

**OUTPUT**

```
enter the no.of elements in the queue= 5
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
 select from 1-4
1
enter the element to be added:4
the queue is=4(0)
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
 select from 1-4
1
enter the element to be added:6
the queue is=4(0)  6(1)
1:enqueue
2:dequeue
3:display
4:exit
```

```
other:invalid
 select from 1-4
1
enter the element to be added:8
the queue is=4(0)  6(1)     8(2)
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
 select from 1-4
2
the element deleted is:4
the queue is=6(1)  8(2)
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
 select from 1-4
1
enter the element to be added:7
the queue is=6(1)  8(2)     7(3)
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
 select from 1-4
1
enter the element to be added:5
the queue is=6(1)  8(2)     7(3)     5(4)
1:enqueue
2:dequeue
3:display
4:exit
other:invalid
select from 1-4
1
enter the element to be added:3
the queue is=6(1)  8(2)     7(3)     5(4)     3(0)
1:enqueue
2:dequeue
3:display
4:exit
```

other:invalid
select from 1-4
4

# PRIORITY QUEUE

## PROGRAM CODE

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>

void main()

{

 int n=10,f=-1,r=-1,ch=0,i;

 int v=0,p=0;

 struct queue

 {

  int val[n];

  int prio[n];

  }q;


  while(ch!=4)

  {

   printf("\nMENU\n\n1.Enque\n2.Deque\n3.Display\n4.Exit");

   printf("\nEnter your choice\t");

   scanf("%d",&ch);


   if(ch==1)

   {

      if(r==n-1)
```

```
{
    printf("QUEUE IS FULL\n");
}

  else if(f==-1&&r==-1)
{
        printf("\nEnter the value \t");
      scanf("%d",&v);
      printf("\nEnter the priority\t");
      scanf("%d",&p);
      f=r=0;
    q.val[r]=v;
    q.prio[r]=p;
}
else
{
    printf("\nEnter the value \t");
      scanf("%d",&v);
      printf("\nEnter the priority\t");
      scanf("%d",&p);

    r++;
    q.val[r]=v;
```

```
        q.prio[r]=p;
  }
}
if(ch==2)
{
  int temp,t;
  for(int i=f;i<r;i++)
  {
      for(int j=0;j<r;j++)
      {
         if(q.prio[j]>q.prio[j+1])
        {
         temp=q.prio[j];
         q.prio[j]=q.prio[j+1];
         q.prio[j+1]=temp;
             t=q.val[j];
         q.val[j]=q.val[j+1];
         q.val[j+1]=t;
        }
      }
   }
        if(f==-1&&r==-1)
```

```
{
   printf("QUEUE IS EMPTY\n");
}
else if(f==r)
{
    printf("\nThe Dequeued element is %d\n",q.val[f]);
  f=r=-1;
}
else
{
  printf("\nThe Dequeued element is %d\n",q.val[f]);
  f++;


}
}
if(ch==3)
{
      if(f==-1&&r==-1)
   printf("QUEUE IS EMPTY\n");
else
{

   printf("\nTHE CURRENT QUEUE IS:\n");
```

```
        printf("Value\tPriority\n");
        for(int i=f;i<r+1;i++)
            printf("%d\t%d\n",q.val[i],q.prio[i]);
      }
       }

      if(ch==4)
      break;

      if(ch>4)
       printf("\nInvalid choice\n");
   }
 }
```

**OUTPUT**

```
MENU

1.Enque
2.Deque
3.Display
4.Exit
Enter your choice  1
Enter the value      4
Enter the priority   2
MENU

1.Enque
2.Deque
3.Display
4.Exit
Enter your choice  1
Enter the value      3
```

```
Enter the priority  1
MENU

1.Enque
2.Deque
3.Display
4.Exit
Enter your choice  1
Enter the value     6
Enter the priority  5
MENU

1.Enque
2.Deque
3.Display
4.Exit
Enter your choice  1
Enter the value     4
Enter the priority  4
MENU

1.Enque
2.Deque
3.Display
4.Exit
Enter your choice  3
THE CURRENT QUEUE IS:
Value   Priority
4    2
3    1
6    5
4    4

MENU

1.Enque
2.Deque
3.Display
4.Exit
Enter your choice  2
The Dequeued element is 3

MENU

1.Enque
```

2.Deque
3.Display
4.Exit
Enter your choice  2
The Dequeued element is 4

MENU

1.Enque
2.Deque
3.Display
4.Exit
Enter your choice  2
The Dequeued element is 4

MENU

1.Enque
2.Deque
3.Display
4.Exit
Enter your choice  2
The Dequeued element is 6

MENU

1.Enque
2.Deque
3.Display
4.Exit
Enter your choice  2
QUEUE IS EMPTY

MENU

1.Enque
2.Deque
3.Display
4.Exit
Enter your choice  4

## DOUBLE-ENDED QUEUE(DEQUEUE)

**PROGRAM CODE**

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
#include<stdlib.h>

int N,front=-1,rear=-1,queue[100];

void enqueueRear(int X)
{
  if((rear+1)%N==front)
     {
       printf("the queue is full");
     }
    else if(front==-1&&rear==-1)
     { front=0;
       rear=0;
       queue[rear]=X;
     }
    else
    {
      rear=(rear+1)%N;
      queue[rear]=X;
    }
}
void enqueueFront(int X)
{
 if((front-1)%N==rear)
    {
     printf("the queue is full");

    }
 else if(front==-1&&rear==-1)
     { front=0;
       rear=0;
       queue[front]=X;
     }
 else if(front==0)
 {front=N-1;
   queue[front]=X;
 }
 else
 {
```

```
 front=(front-1)%N;
 queue[front]=X;
 }
}
int dequeueFront()
{
 if(front==-1&&rear==-1)
  {
    printf("queue is empty");

  }
 else if(front==rear)
 {
  front=-1;
  rear=-1;
  return queue[front+1];
 }
 else
 {
  front=(front+1)%N;
  return queue[front-1];

 }
 }
 int dequeueRear()
{
 if(front==-1&&rear==-1)
  {
    printf("queue is empty");
  }
 else if(front==rear)
 {
  front=-1;
  rear=-1;
  return queue[rear+1];
 }
 else if(rear==0)
 {
  rear=N-1;
  return queue[0];
 }
 else
 {
  rear=(rear-1)%N;
  return queue[rear+1];
```

```
  }
}

void display()
{
  if(front==-1&&rear==-1)
   {
     printf("the queue is empty!");
   }
   else
   {printf("the queue is:");
    int i=front;
    while(i!=rear)
    {
     printf("%d(%d)\t",queue[i],i);
     i=(i+1)%N;
    }
   printf("%d(%d)\t",queue[rear],i);

   }



}


void main()
{

  int choice,X;
  printf("enter the number of elements in double ended queue:");
  scanf("%d",&N);

 while(1)

 {

  printf("enter the below choice\n 1:enter from rear\n2:enter from front\n 3:remove
from front\n4:remove from rear\n5:display\n6:exit\nyour choice:");

  scanf("%d",&choice);

  switch(choice)
  {
    case 1:
```

```
 {
   printf("enter the element to add in rear:");
   scanf("%d",&X);
   enqueueRear(X);
   display();
   break;
 }
 case 2:

{
   printf("enter the element to add in front:");
   scanf("%d",&X);
   enqueueFront(X);
   display();
   break;
 }

 case 3:
 {
   printf("enter the element removed from front: %d",dequeueFront());
   display();
  break;
 }

case 4:
{

 printf("enter the element removed from rear: %d",dequeueRear());
   display();
  break;



}
case 5:
{
  display();
  break;
}
case 6:
{
  exit(0);
}
default:
{
  printf("INVALID");
```

```
  }
  }
  }
}
```

**OUTPUT**

```
enter the number of elements in double ended queue:5
enter the below choice
 1:enter from rear
2:enter from front
 3:remove from front
4:remove from rear
5:display
6:exit
your choice:1
enter the element to add in rear:4
the queue is:4(0)
enter the below choice
 1:enter from rear
2:enter from front
 3:remove from front
4:remove from rear
5:display
6:exit
your choice:1
enter the element to add in rear:6
the queue is:4(0)   6(1)
enter the below choice
 1:enter from rear
2:enter from front
 3:remove from front
4:remove from rear
5:display
6:exit
your choice:2
enter the element to add in front:1
the queue is:1(4)   4(0)     6(1)
enter the below choice
 1:enter from rear
2:enter from front
 3:remove from front
4:remove from rear
5:display
6:exit
your choice:3
```

```
enter the element removed from front: 0
the queue is:4(0)   6(1)
enter the below choice
 1:enter from rear
2:enter from front
 3:remove from front
4:remove from rear
5:display
6:exit
your choice:2
enter the element to add in front:7
the queue is:7(4)   4(0)     6(1)
enter the below choice
 1:enter from rear
2:enter from front
 3:remove from front
4:remove from rear
5:display
6:exit
your choice:4
enter the element removed from rear: 6
the queue is:7(4)   4(0)
enter the below choice
 1:enter from rear
2:enter from front
 3:remove from front
4:remove from rear
5:display
6:exit
your choice:6
```

# LINKED LIST OPERATION

## PROGRAM CODE

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdlib.h>
#include <stdio.h>


void display();
void insert_begin();
void insert_end();
void insert_pos();
void delete_begin();
void delete_end();
void delete_pos();


struct node
{
	int data;
	struct node *next;
};
struct node *head=NULL;
int main()
{
	int choice;
	while(1){

		printf("\n		MENU		\n");
		printf("\n 1.Display	");
		printf("\n 2.Insert at the beginning	");
		printf("\n 3.Insert at the end	");
		printf("\n 4.Insert at specified position	");
		printf("\n 5.Delete from beginning	");
		printf("\n 6.Delete from the end	");
		printf("\n 7.Delete from specified position	");
		printf("\n 8.Exit	");
		printf("\n-------------------------------------");
		printf("Enter your choice:");
		scanf("%d",&choice);
		switch(choice)
		{

			case 1:
				display();
```

```
                        break;
                case 2:
                        insert_begin();
                        display();
                        break;
                case 3:
                        insert_end();
                        display();
                        break;
                case 4:
                        insert_pos();
                        display();
                        break;
                case 5:
                        delete_begin();
                        display();
                        break;
                case 6:
                        delete_end();
                        display();
                        break;
                case 7:
                        delete_pos();
                        display();
                        break;

                case 8:
                        display();
                        exit(0);
                        break;

                default:
                        printf(" Wrong Choice:");
                        break;
            }
        }
        return 0;
}

void display()
{
        struct node *ptr;
        if(head==NULL)
        {
            printf("List is empty:");
```

```
            return;
        }
        else
        {
            ptr=head;
            printf("The List elements are:");
            while(ptr->next!=NULL)
            {
                printf("%d--->",ptr->data );
                ptr=ptr->next ;
            }
            printf("%d--->NULL",ptr->data);
        }
}
void insert_begin()
{
        struct node *temp;
        temp=(struct node *)malloc(sizeof(struct node));
        if(temp==NULL)
        {
            printf("Out of Memory Space:");
            return;
        }
        printf("Enter the data value for the node:" );
        scanf("%d",&temp->data);
        temp->next =NULL;
        if(head==NULL)
        {
            head=temp;
        }
        else
        {
            temp->next=head;
            head=temp;
        }
}
void insert_end()
{
        struct node *temp,*ptr;
        temp=(struct node *)malloc(sizeof(struct node));
        if(temp==NULL)
        {
            printf("Out of Memory Space:");
            return;
        }
```

```
        printf("Enter the data value for the node:" );
        scanf("%d",&temp->data );
        temp->next =NULL;
        if(head==NULL)
        {
                head=temp;
        }
        else
        {
                ptr=head;
                while(ptr->next !=NULL)
                {
                        ptr=ptr->next ;
                }
                ptr->next =temp;
        }
}
void insert_pos()
{
        struct node *ptr,*temp;
        int i,pos;
        temp=(struct node *)malloc(sizeof(struct node));
        if(temp==NULL)
        {
                printf("Out of Memory Space:");
                return;
        }
        printf("Enter the position for the new node to be inserted:");
        scanf("%d",&pos);
        printf("Enter the data value of the node:");
        scanf("%d",&temp->data) ;

        temp->next=NULL;
        if(pos==0)
        {
                temp->next=head;
                head=temp;
        }
        else
        {
                for(i=0,ptr=head;i<pos-1;i++) { ptr=ptr->next;
                        if(ptr==NULL)
                        {
                                printf("Position not found:");
                                return;
```

```
                }
            }
            temp->next =ptr->next ;
            ptr->next=temp;
        }
}
void delete_begin()
{
        struct node *ptr;
        if(ptr==NULL)
        {
            printf("List is Empty:");
            return;
        }
        else
        {
            ptr=head;
            head=head->next ;
            printf("The deleted element is :%d\n",ptr->data);
            free(ptr);
        }
}
void delete_end()
{
        struct node *temp,*ptr;
        if(head==NULL)
        {
            printf("List is Empty:");
            exit(0);
        }
        else if(head->next ==NULL)
        {
            ptr=head;
            head=NULL;
            printf("The deleted element is:%d\n",ptr->data);
            free(ptr);
        }
        else
        {
            ptr=head;
            while(ptr->next!=NULL)
            {
                temp=ptr;
                ptr=ptr->next;
            }
```

```
            temp->next=NULL;
            printf("The deleted element is:%d\n",ptr->data);
            free(ptr);
        }
}
void delete_pos()
{
        int i,pos;
        struct node *temp,*ptr;
        if(head==NULL)
        {
            printf("The List is Empty:");
            exit(0);
        }
        else
        {
            printf("Enter the position of the node to be deleted:");
            scanf("%d",&pos);
            if(pos==0)
            {
                ptr=head;
                head=head->next ;
                printf("The deleted element is:%d\n",ptr->data  );
                free(ptr);
            }
            else
            {
                ptr=head;
                for(i=0;i<pos;i++) { temp=ptr; ptr=ptr->next ;
                    if(ptr==NULL)
                    {
                        printf("Position not Found:");
                        return;
                    }
                }
                temp->next =ptr->next ;
                printf("The deleted element is:%d\n",ptr->data );
                free(ptr);
            }
        }
}
```

**OUTPUT**

```
MENU
1.Display
2.Insert at the beginning
3.Insert at the end
4.Insert at specified position
5.Delete from beginning
6.Delete from the end
7.Delete from specified position
8.Exit
-------------------------------------Enter your choice:3
Enter the data value for the node:6
The List elements are:6--->NULL
            MENU

 1.Display
 2.Insert at the beginning
 3.Insert at the end
 4.Insert at specified position
 5.Delete from beginning
 6.Delete from the end
 7.Delete from specified position
 8.Exit
-------------------------------------Enter your choice:3
Enter the data value for the node:5
The List elements are:6--->5--->NULL
            MENU

 1.Display
 2.Insert at the beginning
 3.Insert at the end
 4.Insert at specified position
 5.Delete from beginning
 6.Delete from the end
 7.Delete from specified position
 8.Exit
-------------------------------------Enter your choice:3
Enter the data value for the node:4
The List elements are:6--->5--->4--->NULL
            MENU

 1.Display
 2.Insert at the beginning
 3.Insert at the end
 4.Insert at specified position
```

```
 5.Delete from beginning
 6.Delete from the end
 7.Delete from specified position
 8.Exit
-------------------------------------Enter your choice:2
Enter the data value for the node:1
The List elements are:1--->6--->5--->4--->NULL
            MENU

 1.Display
 2.Insert at the beginning
 3.Insert at the end
 4.Insert at specified position
 5.Delete from beginning
 6.Delete from the end
 7.Delete from specified position
 8.Exit
-------------------------------------Enter your choice:2
Enter the data value for the node:5
The List elements are:5--->1--->6--->5--->4--->NULL
            MENU

 1.Display
 2.Insert at the beginning
 3.Insert at the end
 4.Insert at specified position
 5.Delete from beginning
 6.Delete from the end
 7.Delete from specified position
 8.Exit
-------------------------------------Enter your choice:4
Enter the position for the new node to be inserted:3
Enter the data value of the node:7
The List elements are:5--->1--->7--->6--->5--->4--->NULL
            MENU

 1.Display
 2.Insert at the beginning
 3.Insert at the end
 4.Insert at specified position
 5.Delete from beginning
 6.Delete from the end
 7.Delete from specified position
 8.Exit
-------------------------------------Enter your choice:7
```

```
Enter the position of the node to be deleted:5
The deleted element is:5
The List elements are:5--->1--->7--->6--->4--->NULL
            MENU

 1.Display
 2.Insert at the beginning
 3.Insert at the end
 4.Insert at specified position
 5.Delete from beginning
 6.Delete from the end
 7.Delete from specified position
 8.Exit
-------------------------------------Enter your choice:6
The deleted element is:4
The List elements are:5--->1--->7--->6--->NULL
            MENU

 1.Display
 2.Insert at the beginning
 3.Insert at the end
 4.Insert at specified position
 5.Delete from beginning
 6.Delete from the end
 7.Delete from specified position
 8.Exit
-------------------------------------Enter your choice:5
The deleted element is :5
The List elements are:1--->7--->6--->NULL
            MENU

 1.Display
 2.Insert at the beginning
 3.Insert at the end
 4.Insert at specified position
 5.Delete from beginning
 6.Delete from the end
 7.Delete from specified position
 8.Exit
-------------------------------------Enter your choice:4
Enter the position for the new node to be inserted:3
Enter the data value of the node:10
The List elements are:1--->7--->10--->6--->NULL
            MENU
```

1.Display
2.Insert at the beginning
3.Insert at the end
4.Insert at specified position
5.Delete from beginning
6.Delete from the end
7.Delete from specified position
8.Exit
----------Enter your choice:8
The List elements are:1--->7--->10--->6--->NULL

# STACK USING LINKED LIST

## PROGRAM CODE

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
#include<stdlib.h>

struct node
{
  int data;
  struct node *next;
};
struct node *top=NULL;



void push(int value)
{
  struct node *newnode;
  newnode= (struct node *)malloc(sizeof(struct node));
  newnode->data=value;
  if(top==NULL)
  {
    newnode->next=NULL;
  }
  else
  {
    newnode->next=top;

  }
  top=newnode;
}

int pop()
{
  struct node *temp;
  temp=top;
  if(top==NULL)
  {printf("stack is empty");
  }

  else
  {
    top=top->next;
    return(temp->data);
```

```
    free(temp);

  }

}

void display()
{
  struct node *ptr;
  printf("stack is:");
 if(top==NULL)
 {
   printf("\nstack is empty");
 }
 else
 {
  ptr=top;
  while(ptr->next!=NULL)
  {

   printf("%d--->",ptr->data);
   ptr=ptr->next;

  }
  printf("%d--->NULL",ptr->data);

 }



}


void main()
{
  int choice,value;

  while(1)
  {
     printf("\nMENU\n1).push\n2).pop\n3).display\n4).exit\nselect your
choice\nchoice:");
     scanf("%d",&choice);

     switch(choice)
     {
```

```
      case 1:
      {
        printf("\nenter the data needed to add in front of stack:");
        scanf("%d",&value);
        push(value);
        display();
        break;
      }

     case 2:
      {
        printf("\n the data deleted from front is %d\n",pop());
        display();
        break;
     }

     case 3:
      {
        display();
        break;
      }
     case 4:
      {display();
       exit(0);

       break;
      }

     default:
      {
       printf("invalid choice");
       break;
      }
   }
 }
}
```

**OUTPUT**

```
MENU
1).push
2).pop
3).display
4).exit
select your choice
choice:1
enter the data needed to add in front of stack:5
```

```
stack is:5--->NULL
MENU
1).push
2).pop
3).display
4).exit
select your choice
choice:1
enter the data needed to add in front of stack:8
stack is:8--->5--->NULL
MENU
1).push
2).pop
3).display
4).exit
select your choice
choice:1
enter the data needed to add in front of stack:3
stack is:3--->8--->5--->NULL
MENU
1).push
2).pop
3).display
4).exit
select your choice
choice:1
enter the data needed to add in front of stack:6
stack is:6--->3--->8--->5--->NULL
MENU
1).push
2).pop
3).display
4).exit
select your choice
choice:2
the data deleted from front is 6
stack is:3--->8--->5--->NULL
MENU
1).push
2).pop
3).display
4).exit
select your choice
choice:2
the data deleted from front is 3
```

```
stack is:8--->5--->NULL
MENU
1).push
2).pop
3).display
4).exit
select your choice
choice:2
the data deleted from front is 8
stack is:5--->NULL
MENU
1).push
2).pop
3).display
4).exit
select your choice
choice:2
the data deleted from front is 5
stack is:
stack is empty
MENU
1).push
2).pop
3).display
4).exit
select your choice
choice:2
stack is empty
 the data deleted from front is 14
stack is:
stack is empty
MENU
1).push
2).pop
3).display
4).exit
select your choice
choice:4
stack is:
stack is empty
```

# QUEUE USING LINKED LIST

## PROGRAM CODE

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
#include<stdlib.h>

struct node
{
  int data;
  struct node *next;
};

struct node *front=NULL;
struct node *rear=NULL;


void enqueue(int value)
{
  struct node *newnode;
  newnode= (struct node*)malloc(sizeof(struct node));
  newnode->data=value;
  newnode->next=NULL;
  if(front==NULL && rear==NULL)
  {

    front=newnode;
    rear=newnode;
  }
 else
 {
   rear->next=newnode;

 }
 rear=newnode;

}

int dequeue()
{
  struct node *temp;

  if(front==NULL)
  {
    printf("queue is empty\n");
```

```
  }
  else
  {
    temp=front;
    front=front->next;
    return temp->data;
    free(temp);

  }

}
void display()
{
  struct node *ptr;
  if(front==NULL)
  {
    printf("queue is empty\n");
  }
  else
  {
    ptr=front;
    printf("the queue is:");
    while(ptr->next!=NULL)
    {
      printf("%d--->",ptr->data);
      ptr=ptr->next;
    }
    printf("%d--->NULL\n",ptr->data);
  }


}



void main()
{
  int choice,value;

  while(1)
  {
     printf("MENU\n1)insert\n2)delete\n3)display\n4)exit\nENTER YOU CHOICE(1-
4)\nchoice:");
     scanf("%d",&choice);
```

```
    switch(choice)
    {
       case 1:
       {
          printf("enter the value to insert in queue:");
          scanf("%d",&value);
          enqueue(value);
          display();
          break;
       }
     case 2:
     {
       printf("the deleted element is:%d\n",dequeue());
       display();
       break;

     }
    case 3:
    {
      display();
      break;
    }
    case 4:
    {
      display();
      exit(0);
   }
   }
  }
}
```

**OUTPUT**

```
MENU
1)insert
2)delete
3)display
4)exit
ENTER YOU CHOICE(1-4)
choice:1
enter the value to insert in queue:5
the queue is:5--->NULL
MENU
1)insert
2)delete
3)display
4)exit
```

```
ENTER YOU CHOICE(1-4)
choice:1
enter the value to insert in queue:7
the queue is:5--->7--->NULL
MENU
1)insert
2)delete
3)display
4)exit
ENTER YOU CHOICE(1-4)
choice:1
enter the value to insert in queue:4
the queue is:5--->7--->4--->NULL
MENU
1)insert
2)delete
3)display
4)exit
ENTER YOU CHOICE(1-4)
choice:1
enter the value to insert in queue:3
the queue is:5--->7--->4--->3--->NULL
MENU
1)insert
2)delete
3)display
4)exit
ENTER YOU CHOICE(1-4)
choice:2
the deleted element is:5
the queue is:7--->4--->3--->NULL
MENU
1)insert
2)delete
3)display
4)exit
ENTER YOU CHOICE(1-4)
choice:2
the deleted element is:7
the queue is:4--->3--->NULL
MENU
1)insert
2)delete
3)display
4)exit
```

ENTER YOU CHOICE(1-4)
choice:2
the deleted element is:4
the queue is:3--->NULL
MENU
1)insert
2)delete
3)display
4)exit
ENTER YOU CHOICE(1-4)
choice:2
the deleted element is:3
MENU
1)insert
2)delete
3)display
4)exit
ENTER YOU CHOICE(1-4)
choice:2
queue is empty
MENU
1)insert
2)delete
3)display
4)exit
ENTER YOU CHOICE(1-4)
choice:3
queue is empty
MENU
1)insert
2)delete
3)display
4)exit
ENTER YOU CHOICE(1-4)
choice:4

# POLYNOMIAL ADDITION USING LINKED LIST

## PROGRAM CODE

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
#include<stdlib.h>

struct node
{
 int coef,expo;
struct node *link;
 };

struct node *Phead,*Qhead,*Rhead;

struct node *ReadPoly();
 void Display(struct node *head);
struct node *AddPoly();



void main()
 {
printf("\nFirst polynomial\n");
 Phead = ReadPoly();

printf("\nSecond polynomial\n");
 Qhead = ReadPoly();

printf("\nFirst polynomial : ");
 Display(Phead);

printf("\nSecond polynomial : ");
Display(Qhead);

 Rhead=AddPoly();
printf("\nResultant Polynomial : ");
 Display(Rhead);

}

 struct node *ReadPoly()
 {
struct node *new,*ptr,*head=NULL;
int n,i;
```

```
printf("\nEnter the total number of terms in the polynomial : ");
scanf("%d",&n);
printf("\nEnter the coefficients and exponents of the polynomial\n");
for(i=1;i<=n;i++)
{
new = (struct node *)malloc(sizeof(struct node));

printf("\nEnter coefficient of term %d : ",i);
 scanf("%d",&new->coef);
 printf("Enter exponent of term %d : ",i);
 scanf("%d",&new->expo);

 new->link=NULL;
 if(head==NULL)
 {
head = new;
ptr = head;
 }
else{
ptr->link = new;
ptr=new;
 }

}

return(head);
 }
 void Display(struct node *head)
 {
 struct node *ptr;

 if(head==NULL)
{
 printf("Polynomial is empty");
 }
 else{
ptr=head;
 while(ptr->link!=NULL)
 {
printf("%dX^%d\t+\t",ptr->coef,ptr->expo);
ptr=ptr->link;
 }

 printf("%dX^%d\t",ptr->coef,ptr->expo);
```

```
 }
}

struct node *AddPoly()
 {
 struct node *new,*P,*Q,*R,*head=NULL;
 P=Phead;
 Q=Qhead;

 while(P!=NULL && Q!=NULL)
 {
if(P->expo==Q->expo)
{

new=(struct node *)malloc(sizeof(struct node));

new->coef=P->coef+Q->coef;

 new->expo=P->expo;

new->link=NULL;

P=P->link;

Q=Q->link;

 }

else if(P->expo > Q->expo)

{
new=(struct node *)malloc(sizeof(struct node));

new->coef=P->coef;
 new->expo=P->expo;
 new->link=NULL;
P=P->link;
 Q=Q->link;
}
else
{
 new=(struct node *)malloc(sizeof(struct node));

new->coef=Q->coef;
new->expo=Q->expo;
```

```
new->link=NULL;
 P=P->link;
Q=Q->link;
}

 if(head==NULL)
 {
 head=new;
 R=head;
 }
else{
 R->link=new;
 R=new;
}
 }

while(P!=NULL)
{

 new=(struct node *)malloc(sizeof(struct node));

 new->expo=P->expo;
new->link=NULL;

 if(head==NULL)
{
head=new;
R=head;
 }
else{
 R->link=new;
 R=new;
 }

P=P->link;
 }

while(Q!=NULL)
{

 new=(struct node *)malloc(sizeof(struct node));

new->coef=Q->coef;
new->expo=Q->expo;
new->link=NULL;
```

```
if(head==NULL)
{
head=new;
 R=head;
 }
else{
 R->link=new;
 R=new;
 }

     Q=Q->link;
 }

 return(head);
}
```

**OUTPUT**

```
First polynomial

Enter the total number of terms in the polynomial : 3

Enter the coefficients and exponents of the polynomial

Enter coefficient of term 1 : 1
Enter exponent of term 1 : 1

Enter coefficient of term 2 : 2
Enter exponent of term 2 : 2

Enter coefficient of term 3 : 3
Enter exponent of term 3 : 3

Second polynomial

Enter the total number of terms in the polynomial : 3

Enter the coefficients and exponents of the polynomial

Enter coefficient of term 1 : 1
Enter exponent of term 1 : 1

Enter coefficient of term 2 : 2
Enter exponent of term 2 : 2
```

Enter coefficient of term 3 : 3
Enter exponent of term 3 : 3

First polynomial : 1X^1 + 2X^2 + 3X^3
Second polynomial : 1X^1 + 2X^2 + 3X^3
Resultant Polynomial : 2X^1 + 4X^2 + 6X^3

# SUM OF TWO VARIABLE POLYNOMIAL ADDITION

## PROGRAM CODE

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
#include<stdlib.h>
struct poly
{
   int coeff;
   int expo1;
   int expo2;
   struct poly *next;
}*newnode,*phead,*qhead,*rhead,*pptr,*qptr,*rptr,*ptr;
void display(struct poly *head)
{
   ptr=head->next;
   while(ptr->next!=NULL)
   {
    if(ptr->expo2==0)
    {
        printf(" %d(x^%d)+",ptr->coeff,ptr->expo1);
    }
    else if(ptr->expo1==0)
    {
        printf(" %d(y^%d)+",ptr->coeff,ptr->expo2);
    }
    else
    {
        printf("%d(x^%d+y^%d)+",ptr->coeff,ptr->expo1,ptr->expo2);
     }
     ptr=ptr->next;
   }
   if(ptr->expo1+ptr->expo2>0)
   {
     if(ptr->expo2==0)
    {
        printf("%d(x^%d)",ptr->coeff,ptr->expo1);
    }
    else if(ptr->expo1==0)
    {
        printf("%d(y^%d)",ptr->coeff,ptr->expo2);
    }
    else
    {
```

```
            printf("%d(x^%d+y^%d)",ptr->coeff,ptr->expo1,ptr->expo2);
        }
    }
    else
        printf("%d",ptr->coeff);
    printf("\n");
}
struct poly * insert(int co,int ex1,int ex2)
{
    newnode=(struct poly*)malloc(sizeof(struct poly));
    newnode->coeff=co;
    newnode->expo1=ex1;
    newnode->expo2=ex2;
    newnode->next=NULL;
    return(newnode);
}
void main()
{
    int m=0,n=0,i=0,ex1=0,ex2=0,co=0;
    phead=(struct poly*)malloc(sizeof(struct poly));
    pptr=phead;
    phead->next=NULL;
    phead->expo1=0;
    phead->expo2=0;
    phead->coeff=0;
    printf("enter the number of terms in first polynomial: ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("enter the power of x: ");
      scanf("%d",&ex1);
      printf("enter the power of y: ");
      scanf("%d",&ex2);
      printf("enter the coefficient: ");
      scanf("%f",&co);
      newnode=insert(co,ex1,ex2);
        pptr->next=newnode;
        pptr=newnode;

    }
    qhead=(struct poly*)malloc(sizeof(struct poly));
    qptr=qhead;
    qhead->next=NULL;
    qhead->expo1=0;
    qhead->expo2=0;
```

```
qhead->coeff=0;
printf("enter the number of terms in second polynomial: ");
scanf("%d",&m);
for(i=0;i<m;i++)
{
   printf("enter the power of x: ");
 scanf("%d",&ex1);
 printf("enter the power of y: ");
 scanf("%d",&ex2);
 printf("enter the coefficient: ");
 scanf("%f",&co);
   newnode=insert(co,ex1,ex2);
   qptr->next=newnode;
   qptr=newnode;
}
printf("\nthe first polynomial is: ");
display(phead);
printf("\nthe second polynomial is: ");
display(qhead);
pptr=phead->next;
qptr=qhead->next;
rhead=(struct poly*)malloc(sizeof(struct poly));
rhead->next=NULL;
rhead->expo1=0;
rhead->expo2=0;
rhead->coeff=0;
rptr=rhead;
while(pptr!=NULL&&qptr!=NULL)
{
   if(pptr->expo1==qptr->expo1&&pptr->expo2==qptr->expo2)
   {
      rptr->next=insert(pptr->coeff+qptr->coeff,pptr->expo1,pptr->expo2);
      rptr=rptr->next;
      pptr=pptr->next;
      qptr=qptr->next;
   }
   else
   {
      if(pptr->expo1+pptr->expo2==qptr->expo1+qptr->expo2)
      {
           if(pptr->expo1>qptr->expo1)
           {
                rptr->next=insert(pptr->coeff,pptr->expo1,pptr->expo2);
                     rptr=rptr->next;
                     pptr=pptr->next;
```

```
            }
            else
            {
                rptr->next=insert(qptr->coeff,qptr->expo1,qptr->expo2);
                rptr=rptr->next;
                qptr=qptr->next;
            }
        }
        else
        {
            if(pptr->expo1+pptr->expo2>qptr->expo1+qptr->expo2)
            {
                rptr->next=insert(pptr->coeff,pptr->expo1,pptr->expo2);
                    rptr=rptr->next;
                    pptr=pptr->next;
                }
                else
                {
                    rptr->next=insert(qptr->coeff,qptr->expo1,qptr->expo2);
                rptr=rptr->next;
                qptr=qptr->next;
            }
        }
      }
   }
   while(pptr!=NULL)
   {
      rptr->next=insert(pptr->coeff,pptr->expo1,pptr->expo2);
      rptr=rptr->next;
      pptr=pptr->next;
   }
   while(qptr!=NULL)
   {
      rptr->next=insert(qptr->coeff,qptr->expo1,qptr->expo2);
      rptr=rptr->next;
      qptr=qptr->next;
   }
   printf("\nthe sum of polynomials is: ");
   display(rhead);
}
```

**OUTPUT**

```
enter the number of terms in first polynomial: 3
enter the power of x: 2
enter the power of y: 3
enter the coefficient: 1
```

enter the power of x: 2
enter the power of y: 3
enter the coefficient: 5
enter the power of x: 1
enter the power of y: 2
enter the coefficient: 7
enter the number of terms in second polynomial: 2
enter the power of x: 2
enter the power of y: 3
enter the coefficient: 66
enter the power of x: 1
enter the power of y: 2
enter the coefficient: 24

the first polynomial is: 1.00x^2y^3+5.00x^2y^3+7.00x^1y^2

the second polynomial is: 66.00x^2y^3+24.00x^1y^2

the sum of polynomials is: 67.00x^2y^3+5.00x^2y^3+31.00x^1y^2

## STUDENTS DETAILS

**PROGRAM CODE**

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
#include<stdlib.h>

struct node
{
  int number;
  char name[100];
  int mark;
  struct node *next;
};

struct node *head=NULL;
struct node *end=NULL;

void insert()
{
  struct node *newnode;
  newnode=(struct node *)malloc(sizeof(struct node));
  printf("ENTER THE ROLL NUMBER OF STUDENT:");
  scanf("%d",&newnode->number);
  printf("ENTER THE NAME OF CHILD:");
  scanf("%s",newnode->name);
  printf("ENTER THE MARK OF STUDENT:");
  scanf("%d",&newnode->mark);
  newnode->next=NULL;
  if(head==NULL && end==NULL)
  {
    head=newnode;
    end=newnode;

  }
  else
  {
   end->next=newnode;
   end=newnode;
  }

}

void delete()
```

```
{
  struct node *temp,*ptr;
  int flag=0,x;
  temp=head;
  printf("ENTER THE ROLL NUMBER OF STUDENT TO BE DELETED: ");
  scanf("%d",&x);
  if(head->number==x)
  {
   head=head->next;
   free(temp);
   flag=1;
  }
 else
 {
  while(temp->next!=NULL)
   {
     if(temp->next->number==x)
     {
       ptr=temp->next;
       if(ptr->next==NULL)
        {
        end=temp;
        }
       temp->next=ptr->next;
       free(ptr);
       flag=1;

     }

    temp=temp->next;
     }
  }
 if(flag==0)
 {
  printf("ITEM NOT FOUND IN THE LIST %d",x);
 }

}
void search()
{
 struct node *temp;
 int flag=0,y;
 temp=head;
 printf("ENTER THE ROLL NUMBER OF STUDENT WHOSE MARK IS
NEEDED TO SEARCH");
```

```
 scanf("%d",&y);
 while(temp!=NULL)
 {
   if(temp->number==y)
   {
     printf("the student details:\n");
     printf("NAME:%s\nROLL
NO:%d\nMARK:%d",temp->name,temp->number,temp->mark);
     flag=1;

   }
  temp=temp->next;
 }
 if(flag==0)
 {
  printf("ITEM IS NOT FOUND IN LIST %d",y);
 }


}

void sort()
{
 struct node *temp1,*temp2;
 int i,n,m;
 char a[100];
 temp1=head;
 temp2=head;
 for(temp1=head;temp1->next!=NULL;temp1=temp1->next)
 {
   for(temp2=head;temp2->next!=NULL;temp2=temp2->next)
    {
      if((temp2->number) > (temp2->next->number) )
       {
        n=temp2->number;
        m=temp2->mark;
        for(i=0;temp2->name[i]!='\0';i++)
        {
          a[i]=temp2->name[i];

        }
        a[i]='\0';

       temp2->number=temp2->next->number;
       temp2->mark=temp2->next->mark;
```

```
        temp2->next->number=n;
        temp2->next->mark=m;
        for(i=0;temp2->next->name[i]!='\0';i++)
        {
          temp2->name[i]=temp2->next->name[i];

        }
        temp2->name[i]='\0';
        for(i=0;a[i]!='\0';i++)
        {
         temp2->next->name[i]=a[i];
        }
        }
     }
 }
}

void display()
{
 struct node *temp;
 temp=head;
 if(head==NULL)
 {
  printf("LIST IN EMPTY");
 }
 else
 {
  while (temp!=NULL)
  {
   printf("\nthe student details:\n");
   printf("NAME:%s\nROLL
NO:%d\nMARK:%d",temp->name,temp->number,temp->mark);
   temp=temp->next;
  }
 }
}
int main()
{
  int choice;

  while(1)
  {
    printf("\nMENU\n1:insert\n2:delete\n3:search\n4:sort\nSELECT 1/2/3/4");
    scanf("%d",&choice);
    switch (choice)
```

```
    {
     case 1:
     {
       insert();
       display();
       break;
     }
     case 2:
     {
      delete();
      display();
      break;
     }
    case 3:
    {
     search();
     break;
    }
    case 4:
    {
     sort();
     display();
     break;

    }
    case 5:
    {
    display();
    exit(0);
    break;
    }
   }


  }



}
```

**OUTPUT**

```
MENU
1:insert
2:delete
3:search
4:sort
```

SELECT 1/2/3/4 1
1
ENTER THE ROLL NUMBER OF STUDENT:4
ENTER THE NAME OF CHILD:RONALDO
ENTER THE MARK OF STUDENT:95
the student details:
NAME:RONALDO
ROLL NO:4
MARK:95
MENU
1:insert
2:delete
3:search
4:sort
SELECT 1/2/3/4 1
1
ENTER THE ROLL NUMBER OF STUDENT:1
ENTER THE NAME OF CHILD:SANIN
ENTER THE MARK OF STUDENT:98
the student details:
NAME:RONALDO
ROLL NO:4
MARK:95
the student details:
NAME:SANIN
ROLL NO:1
MARK:98
MENU
1:insert
2:delete
3:search
4:sort
SELECT 1/2/3/4 1
1
ENTER THE ROLL NUMBER OF STUDENT:3
ENTER THE NAME OF CHILD:APARNA
ENTER THE MARK OF STUDENT:88
the student details:
NAME:RONALDO
ROLL NO:4
MARK:95
the student details:
NAME:SANIN
ROLL NO:1
MARK:98

the student details:
NAME:APARNA
ROLL NO:2
MARK:88
MENU
1:insert
2:delete
3:search
4:sort
SELECT 1/2/3/4 1
1
ENTER THE ROLL NUMBER OF STUDENT:3
ENTER THE NAME OF CHILD:MESSI
ENTER THE MARK OF STUDENT:78
the student details:
NAME:RONALDO
ROLL NO:4
MARK:95
the student details:
NAME:SANIN
ROLL NO:1
MARK:98
the student details:
NAME:APARNA
ROLL NO:2
MARK:88
the student details:
NAME:MESSI
ROLL NO:3
MARK:78
MENU
1:insert
2:delete
3:search
4:sort
SELECT 1/2/3/4 3
3
ENTER THE ROLL NUMBER OF STUDENT WHOSE MARK IS NEEDED TO SEARCH 4
4
the student details:
NAME:RONALDO
ROLL NO:4
MARK:95
MENU

1:insert
2:delete
3:search
4:sort
SELECT 1/2/3/4 4
4

the student details:
NAME:SANIN
ROLL NO:1
MARK:98
the student details:
NAME:APARNA
ROLL NO:2
MARK:88
the student details:
NAME:MESSI
ROLL NO:3
MARK:78
the student details:
NAME:RONALDO
ROLL NO:4
MARK:95
MENU
1:insert
2:delete
3:search
4:sort
SELECT 1/2/3/4

# REVERSING QUEUE USING STACK

## PROGRAM CODE

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
#include<stdlib.h>

struct node
{
  int data;
  struct node *next;

};


struct node *front=NULL;
struct node *rear=NULL;
struct node *top=NULL;

void enqueue(int item)
{
  struct node *newnode;
  newnode= (struct node*)malloc(sizeof(struct node));
  newnode->data=item;
  newnode->next=NULL;
  if(front==NULL)
  {

    front=newnode;
    rear=newnode;

  }
 else
 {
   rear->next=newnode;

 }

 rear=newnode;

}

void dequeue()
```

```
{ struct node *temp,*new,*temp1;
 if(front==NULL)
 {
   printf("queue is empty");
 }
else
{ printf("dequeued is: %d\n",front->data);
  temp=front;
  new=(struct node *)malloc(sizeof(struct node));
  new->data=temp->data;
  new->next=top;
  top=new;
  front=front->next;
  free(temp);

}
temp1=top;
printf("the stack is:");
while(temp1->next!=NULL)
{
 printf("%d--->",temp1->data);
 temp1=temp1->next;
}
printf("%d--->NULL\n",temp1->data);
}

void display()
{
  struct node *temp;
  temp=front;
  if(front==NULL)
  {
   printf("queue is empty");
  }
 else
 { printf("the queue is:");
  while(temp->next!=NULL)
  {
   printf("%d--->",temp->data);
   temp=temp->next;

  }
  printf("%d-->NULL\n",temp->data);
 }
```

```
}

void rev()
{
  struct node *temp;
  temp=top;
  while(temp!=NULL)
  {
   enqueue(temp->data);
   temp=temp->next;

  }
   display();

}

void main()
{
int choice,item;
while(1)
  {
     printf("MENU\n1)insert\n2)delete\n3)display\n4)reverse\nENTER YOU
CHOICE(1-4)\nchoice:");
     scanf("%d",&choice);

     switch(choice)
     {
        case 1:
        {
           printf("enter the value to insert in queue:");
           scanf("%d",&item);
           enqueue(item);
           display();
           break;
        }
      case 2:
      {
        dequeue();
        display();
        break;

      }
     case 3:
     {
       display();
```

```
      break;
     }
     case 4:
     {
      rev();
      exit(0);
      break;
     }
 }
  }
}
```

**OUTPUT**

```
MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:1
enter the value to insert in queue:3
the queue is:3-->NULL
MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:1
enter the value to insert in queue:11
the queue is:3--->11-->NULL
MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:1
enter the value to insert in queue:5
the queue is:3--->11--->5-->NULL
MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
```

```
choice:1
enter the value to insert in queue:7
the queue is:3--->11--->5--->7-->NULL
MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:1
enter the value to insert in queue:3
the queue is:3--->11--->5--->7--->3-->NULL
MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:1
enter the value to insert in queue:9
the queue is:3--->11--->5--->7--->3--->9-->NULL
MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:1
enter the value to insert in queue:8
the queue is:3--->11--->5--->7--->3--->9--->8-->NULL
MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:2
dequeued is: 3
the stack is:3--->NULL
the queue is:11--->5--->7--->3--->9--->8-->NULL
MENU
1)insert
2)delete
3)display
4)reverse
```

ENTER YOU CHOICE(1-4)
choice:2
dequeued is: 11
the stack is:11--->3--->NULL
the queue is:5--->7--->3--->9--->8-->NULL
MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:2
dequeued is: 5
the stack is:5--->11--->3--->NULL
the queue is:7--->3--->9--->8-->NULL
MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:2
dequeued is: 7
the stack is:7--->5--->11--->3--->NULL
the queue is:3--->9--->8-->NULL
MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:2
dequeued is: 3
the stack is:3--->7--->5--->11--->3--->NULL
the queue is:9--->8-->NULL
MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:2
dequeued is: 9
the stack is:9--->3--->7--->5--->11--->3--->NULL
the queue is:8-->NULL

MENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:2
dequeued is: 8
the stack is:8--->9--->3--->7--->5--->11--->3--->NULL
queue is emptyMENU
1)insert
2)delete
3)display
4)reverse
ENTER YOU CHOICE(1-4)
choice:4
the queue is:8--->9--->3--->7--->5--->11--->3-->NULL

# CHECKING PALINDROME USING LINKED LIST

## PROGRAM CODE

```
//SANIN MOHAMMED N
//B21CSB55
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
struct node
{
 struct node *prev;
struct node *next;
char data;
};
struct node *head;
void insertion_last(char x);
void display();

void main ()
{
char str[20];
int flag=0;
printf("Enter the string\n");
scanf("%s",str);
int n = strlen(str);
int i;
for(i=0;i<n;i++)
{
insertion_last(str[i]);
}

struct node *p=head;
struct node *tail;

while(p->next!=NULL)
{
 p=p->next;
}
tail=p;
p=head;

for(i=0;i<n/2;i++)
```

```
{
if(tolower(p->data)!= tolower(tail->data))
{
flag=1;
break;
}
else{
p=p->next;
tail=tail->prev;
}
}

if(flag==1)
{
printf("NOT PALINDROME\n");
}
else{
printf("PALINDROME\n");
}
}
void insertion_last(char x)
{
struct node *ptr,*temp;
int item;
ptr = (struct node *) malloc(sizeof(struct node));
if(ptr == NULL)
{
printf("\nOVERFLOW");
}
else
{
ptr->data=x;
if(head == NULL)
{
ptr->next = NULL;
ptr->prev = NULL;
head = ptr;
}
else
{
 temp = head;
while(temp->next!=NULL)
{
 temp = temp->next;
}
```

```
temp->next = ptr;
 ptr ->prev=temp;
 ptr->next = NULL;
}

}
}
```

**OUTPUT**

```
Enter the string
MAlayALAm
PALINDROME

Enter the string
APPLE
NOT PALINDROME
```