Queue contents are:10 15 20 25

6.Implement a program to multiply two polynomials using singly linked list.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
int count;
struct node
{
   int co,po;
   struct node *addr;
};
typedef struct node *NODE;
NODE insertend(NODE start,int co,int po)
{
   NODE temp,cur;
   temp=(NODE)malloc(sizeof(struct node));
   temp->co=co;
   temp->po=po;
   temp->addr=NULL;
   if(start==NULL)
      return temp;
   cur=start;
   while(cur->addr!=NULL)
      cur=cur->addr;
   cur->addr=temp;
   return start;
}

void display(NODE start)
{
```

```
    NODE temp;
    if(start==NULL)
        printf("\n Polynomial Empty");
    else
    {
        temp=start;
        while(temp->addr!=NULL)
        {
            printf("%dx^%d+",temp->co,temp->po);
            temp=temp->addr;
        }
            printf("%dx^%d\n",temp->co,temp->po);
    }
}

NODE addterm(NODE res,int co,int po)
{
    NODE temp,cur;
    temp=(NODE)malloc(sizeof(struct node));
    temp->co=co;
    temp->po=po;
    temp->addr=NULL;
    if(res==NULL)
        return temp;
    cur=res;
    while(cur!=NULL)
    {
        if(cur->po==po)
        {
            cur->co=cur->co+co;
```

```
            return res;
        }
        cur=cur->addr;
    }
    if(cur==NULL)
        res=insertend(res,co,po);
    return res;
}
NODE multiply(NODE poly1,NODE poly2)
{
    NODE p1,p2,res=NULL;
    for(p1=poly1;p1!=NULL;p1=p1->addr)
        for(p2=poly2;p2!=NULL;p2=p2->addr)
            res=addterm(res,p1->co*p2->co,p1->po+p2->po);
    return res;
}
int main()
{
    NODE poly1=NULL,poly2=NULL,poly;
    int co,po;
    int i,n,m;
    printf("\nRead no of terms of first polynomial:");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        printf("\n Read CO and PO of %d term : ",i);
        scanf("%d%d",&co,&po);
        poly1=insertend(poly1,co,po);
    }
    printf("\n First polynomial is\n");
```

```
    display(poly1);
    printf("\nRead no of terms of second polynomial:");
    scanf("%d",&m);
    for(i=1;i<=m;i++)
    {
       printf("\n Read CO and PO of %d term : ",i);
       scanf("%d%d",&co,&po);
       poly2=insertend(poly2,co,po);
    }
    printf("\n Second polynomial is\n");
    display(poly2);
    poly=multiply(poly1,poly2);
    printf("\n Resultant polynomial is\n");
    display(poly);
    return 0;

}
```

OUTPUT:

```
Read the no.of terms in polynomil1:3
Read the co-efficient and power of 1 term:4 2
Read the co-efficient and power of 2 term:2 1
Read the co-efficient and power of 1 term:1 0
Read the no.of terms in polynomil1:2
Read the co-efficient and power of 2 term:3 1
Read the co-efficient and power of 2 term:1 0

First polynomial is:4*x^2+2*x^1+1*x^0
Second polynomial is:3*x^1+1*x^0

Resultant polynomial is:12*x^3+10*x^2+5*x^2+5*x^1+1*x^0
```

7.Design a doubly linked list to represent sparse matrix. Each node in the list can have the row and column index of the matrix element and the value

of the element. Print the complete matrix as the output.

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

struct node
{
   int row,col,data;
   struct node *next;
   struct node *prev;
};
typedef struct node *NODE;

NODE insertend(NODE start,int row,int col,int item)
{
   NODE temp,cur;
   temp=(NODE)malloc(sizeof(struct node));
   temp->row=row;
   temp->col=col;
   temp->data=item;
   temp->next=NULL;
   temp->prev=NULL;
   if(start == NULL)
      return temp;
   cur=start;
   while(cur->next!=NULL)
      cur = cur->next;
   cur->next=temp;
   temp->prev=cur;
```

```
    return start;
}

void display(NODE start)
{
    NODE temp;
    if(start==NULL)
        printf("\n list is empty");
    else
    {
        printf("\nROW\tCOL\tDATA\n");
        temp=start;
        while(temp!=NULL)
        {
            printf("%d\t%d\t%d\n",temp->row,temp->col,temp->data);
            temp=temp->next;
        }
    }
}

void displaymatrix(NODE start,int m,int n)
{
    NODE temp;
    int i,j;
    temp=start;
    printf("\n The Sparse matrix is\n");
    for(i=1;i<=m;i++)
        {
            for(j=1;j<=n;j++)
            {
```

```
            if(temp!=NULL && temp->row == i && temp->col == j)
            {
               printf("%d\t",temp->data);
               temp=temp->next;
            }
            else
               printf("0\t");
      }
     printf("\n");
    }

}



int main()
{
   NODE start = NULL;
   int i,j,m,n,item;
   printf("\n Read the order of the matrix\n");
   scanf("%d%d",&m,&n);
   printf("\n Read the matrix\n");
   for(i=1;i<=m;i++)
   {
      for(j=1;j<=n;j++)
      {
         scanf("%d",&item);
         if(item!=0)
            start=insertend(start,i,j,item);
       }
   }
```

```
    display(start);
    displaymatrix(start,m,n);
    return 0;
}
```

OUTPUT:

```
Read the order of the matrix:3 4
Read the elements of Matrix:
0
0
2
0
0
6
5
0
3
0
0
0
ROW COL DATA
1  3  2
2  3  6
2  4  5
3  1  1

THE SPARSE MATRIX IS:
0 0 2 0
0 6 5 0
3 0 0 0
```

8.Write a C program to create Binary Tree and to traverse the tree using In-order, Preorder and Post order.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *left;
   struct node *right;
};
typedef struct node *NODE;

NODE create_node(int item)
{
   NODE temp;
   temp=(NODE)malloc(sizeof(struct node));
   temp->data=item;
   temp->left=NULL;
   temp->right=NULL;
   return temp;
}

NODE Insertbst(NODE root,int item)
{
   NODE temp;
   temp=create_node(item);
   if(root==NULL)
      return temp;
   else
```

```
    {
      if(item < root->data)
          root->left=Insertbst(root->left,item);
       else
          root->right=Insertbst(root->right,item);
    }
    return root;

}

void preorder(NODE root)
{
    if(root!=NULL)
    {
        printf("%d\t",root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

void inorder(NODE root)
{
    if(root!=NULL)
    {
        inorder(root->left);
        printf("%d\t",root->data);
        inorder(root->right);
    }
}
```

```
void postorder(NODE root)
{
   if(root!=NULL)
   {
      postorder(root->left);
      postorder(root->right);
      printf("%d\t",root->data);
   }
}


int main()
{
   NODE root = NULL;
   int ch,item;
   for(;;)
   {
      printf("\n 1. Insert");
      printf("\n 2. Preorder");
      printf("\n 3. Inorder");
      printf("\n 4. Postorder");
      printf("\n 5. Exit");
      printf("\n Read ur choice:");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:printf("\n Read element to be inserted :");
            scanf("%d",&item);
            root=Insertbst(root,item);
            break;
```

```
            case 2:printf("\n The Preorder traversal is\n");
                preorder(root);
                break;
            case 3:printf("\n The Inorder traversal is\n");
                inorder(root);
                break;
            case 4:printf("\n The Postorder traversal is\n");
                postorder(root);
                break;
            default :exit(0);
        }
    }
    return 0;
}
```

OUTPUT:

```
1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit
Read your choice:1
Read the element to be inserted:18
1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit
Read your choice:1
Read the element to be inserted:4
1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit
Read your choice:1
Read the element to be inserted:1
1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:1
Read the element to be inserted:0
1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit
```

Read your choice:1

Read the element to be inserted:7

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:1

Read the element to be inserted:12

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:1

Read the element to be inserted:47

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:1

Read the element to be inserted:21

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:1

Read the element to be inserted:25

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:1

Read the element to be inserted:90

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:2

The Pre-order traversal is:

18,4,1,0,7,12,47,25,21,90

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:3

The Inorder traversal is:

0,1,4,7,12,18,21,25,47,90

Read your choice:4

The Post order traversal is:

0,12,7,1,25,21,90,47,4,18

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:5

9.Write a C program to implement priority queue using Heap.

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

void heapify(int a[10],int n)
{
   int i,k,v,j,flag=0;
   for(i=n/2;i>=1;i--)
   {
      k=i;
      v=a[k];
      while(!flag && 2*k <= n)
      {
         j=2*k;
         if(j<n)
         {
            if(a[j]<a[j+1])
               j=j+1;
         }
         if(v>=a[j])
            flag=1;
         else
         {
            a[k]=a[j];
            k=j;
         }
      }
      a[k]=v;
      flag=0;
```

```
    }
}

int main()
{
    int n,i,a[10],ch;
    for(;;)
    {
        printf("\n 1. Create Heap");
        printf("\n 2. Extractmax");
        printf("\n 3. Exit");
        printf("\n Read Choice :");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:printf("\n Read no of elements :");
                   scanf("%d",&n);
                   printf("\n Read Elements\n");
                   for(i=1;i<=n;i++)
                    scanf("%d",&a[i]);
                   heapify(a,n);
                   printf("\n Elements after heap\n");
                   for(i=1;i<=n;i++)
                    printf("%d\t",a[i]);
                    break;
            case 2:if(n>=1)
                   {
                      printf("\n Element deleted is %d\n",a[1]);
                      a[1]=a[n];
                      n=n-1;
```

```
                heapify(a,n);
                if(n!=0)
                {
                   printf("\n Elements after reconstructing heap\n");
                   for(i=1;i<=n;i++)
                      printf("%d\t",a[i]);
                }
             }
             else
                printf("\n No element to delete");
             break;
       default:exit(0);

     }
   }
   return 0;
}
```

OUTPUT:

1.Create Heap 2.Extract Max 3.Exit

Read Choice:1

Read no.of elements:5

Elements after heap:

40 30 15 10 20

1.Create Heap 2.Extract Max 3.Exit

Read Choice:2

Element deleted is 40

Elements after reconstructing heap:

30 20 15 10

1.Create Heap 2.Extract Max 3.Exit

Read Choice:3

10.Write a C program to implement Hashing using Linear probing. Implement insertion, deletion, search and display.

```
#include <stdio.h>
#include<stdlib.h>
#define TABLE_SIZE 10

int h[TABLE_SIZE]={NULL};

void insert()
{
 int key,index,i,flag=0,hkey;
 printf("\nenter a value to insert into hash table:");
 scanf("%d",&key);
 hkey=key%TABLE_SIZE;
 for(i=0;i<TABLE_SIZE;i++)
   {
    index=(hkey+i)%TABLE_SIZE;
    if(h[index] == NULL)
    {
      h[index]=key;
       break;
    }
   }
   if(i == TABLE_SIZE)
      printf("\nelement cannot be inserted\n");
}

void search()
{
```

```
 int key,index,i,flag=0,hkey;
 printf("\nenter search element:");
 scanf("%d",&key);
 hkey=key%TABLE_SIZE;
 for(i=0;i<TABLE_SIZE; i++)
 {
   index=(hkey+i)%TABLE_SIZE;
   if(h[index]==key)
   {
     printf("value is found at index %d",index);
     break;
   }
 }
 if(i == TABLE_SIZE)
   printf("\n value is not found\n");
}

void display()
{
 int i;
 printf("\nelements in the hash table are \n");
 for(i=0;i< TABLE_SIZE; i++)
     printf("\nat index %d \t value =  %d",i,h[i]);

}
main()
{
   int ch,i;
   for(;;)
   {
```

```
        printf("\n1.Insert\n2.Display\n3.Search\n4.Exit\n");
        printf("\n Read Choice :");
        scanf("%d",&ch);
        switch(ch)
        {
          case 1:
             insert();
             break;
          case 2:
             display();
             break;
          case 3:
             search();
             break;
          default:exit(0);
        }
      }
}
```

OUTPUT:

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

Read choice:1

Enter the value in the hash table:1

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

Read choice:1

Enter the value in the hash table:44

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

Read choice:1

Enter the value in the hash table:67

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

Read choice:1

Enter the value in the hash table:22

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

Read choice:1

Enter the value in the hash table:58

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

Read choice:1

Enter the value in the hash table:64

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

Read choice:2

Elements in the hash table are

At index 0 value = 0

At index 0 value = 0

At index 0 value = 12

At index 0 value = 24

At index 0 value = 44

At index 0 value = 64

At index 0 value = 0

At index 0 value = 67

At index 0 value = 58

At index 0 value = 0

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

Read choice:3

Enter the search element:44

Value is found at index 4

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

Read choice:4