Progressive Education Society's

Modern College of Engineering, Pune

MCA Department

A.Y.2022-23

(310902) Data Structure and Algorithm Laboratory

****

Class: FY-MCA Shift /Div: A Batch: F2 Roll Number: 51043

Name: Vanessa Reetu Prashant More Assignment No: 1 Date of Implementation: 15/12/2022

****

Q1.Write a program to find mean of n numbers using arrays.

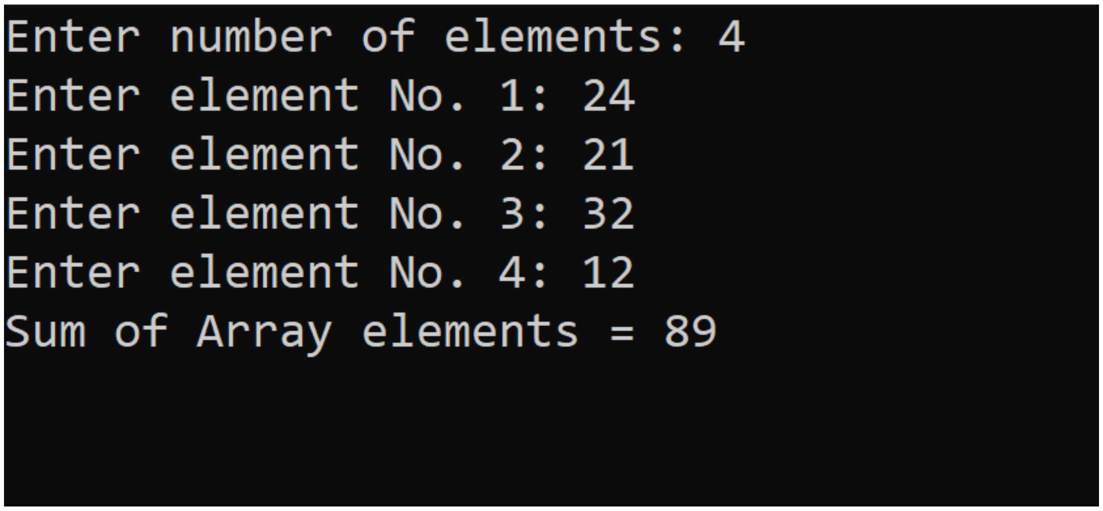
Program:

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

int main(){
        int num, i, max,sum=0;
        int arr[100];
        //clrscr();

        printf("Enter number of elements: ");
        scanf("%d",&num);

                for(i=0;i<num;i++){
                        printf("Enter element No. %d: ",i+1);
                        scanf("%d",&arr[i]);
                }

                for(i=0; i<num; i++){
                        sum = sum + arr[i];
                }

        printf("Sum of Array elements = %d ",sum);

        getch();
        return 0;
}
```

Output:

```
Enter number of elements: 4
Enter element No. 1: 24
Enter element No. 2: 21
Enter element No. 3: 32
Enter element No. 4: 12
Sum of Array elements = 89
```

In Editor:



```
Enter number of elements: 4
Enter element No. 1: 24
Enter element No. 2: 21
Enter element No. 3: 32
Enter element No. 4: 12
Sum of Array elements = 89
```

Q2. Write a program to interchange the smallest & largest no of n numbers using arrays.

Program:

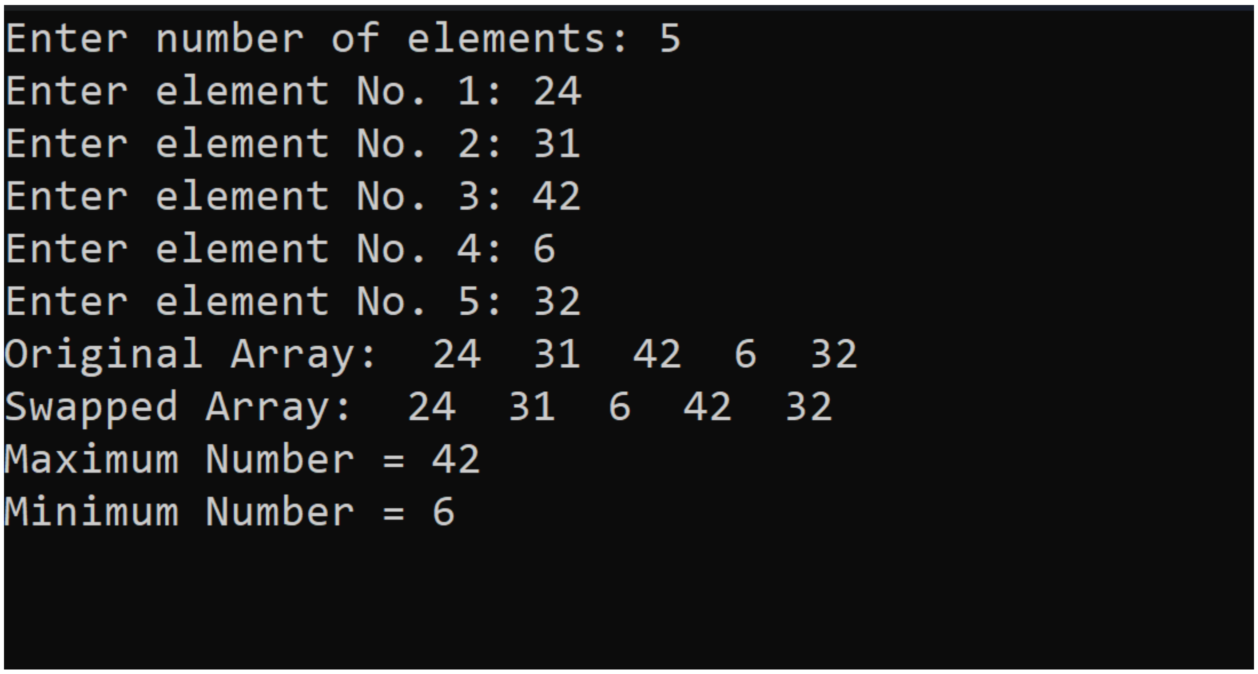
```
#include<stdio.h>
#include<conio.h>

int main(){
int num, i, max, x, y, min;
int arr[100];
//clrscr();

    printf("Enter number of elements: ");
    scanf("%d",&num);

            for(i=0;i<num;i++){
                printf("Enter element No. %d: ",i+1);
                scanf("%d",&arr[i]);
            }

            max =arr[0];

            for(i=0; i<num; i++){
                if(max < arr[i]){
                    max = arr[i];
                    y = i;
                }
            }

            min = arr[0];
            for(i=0; i<num; i++){
                if(min > arr[i]){
                    min = arr[i];
                    x = i;
                }
            }

            printf("Original Array: ");
            for(i=0;i<num;i++){
                printf(" %d ",arr[i]);
            }



            for(i=0; i<num; i++){
                if(i==y){
                  arr[i]=min;
                }
                if(i==x){
                  arr[i]=max;
                }
            }

            printf("\nSwapped Array: ");
```

```
            for(i=0;i<num;i++){
                printf(" %d ",arr[i]);
            }

            printf("\nMaximum Number = %d ",max);
            printf("\nMinimum Number = %d ",min);

            getch();
            return 0;
}
```

Output:

```
Enter number of elements: 5
Enter element No. 1: 24
Enter element No. 2: 31
Enter element No. 3: 42
Enter element No. 4: 6
Enter element No. 5: 32
Original Array:  24  31  42  6  32
Swapped Array:  24  31  6  42  32
Maximum Number = 42
Minimum Number = 6
```

In Editor:



```
Enter number of elements: 5
Enter element No. 1: 24
Enter element No. 2: 31
Enter element No. 3: 42
Enter element No. 4: 6
Enter element No. 5: 32
Original Array:  24  31  42  6  32
Swapped Array:  24  31  6  42  32
Maximum Number = 42
Minimum Number = 6
```

Q3. Write a program to delete a no from an array which is already sorted in ascending order.

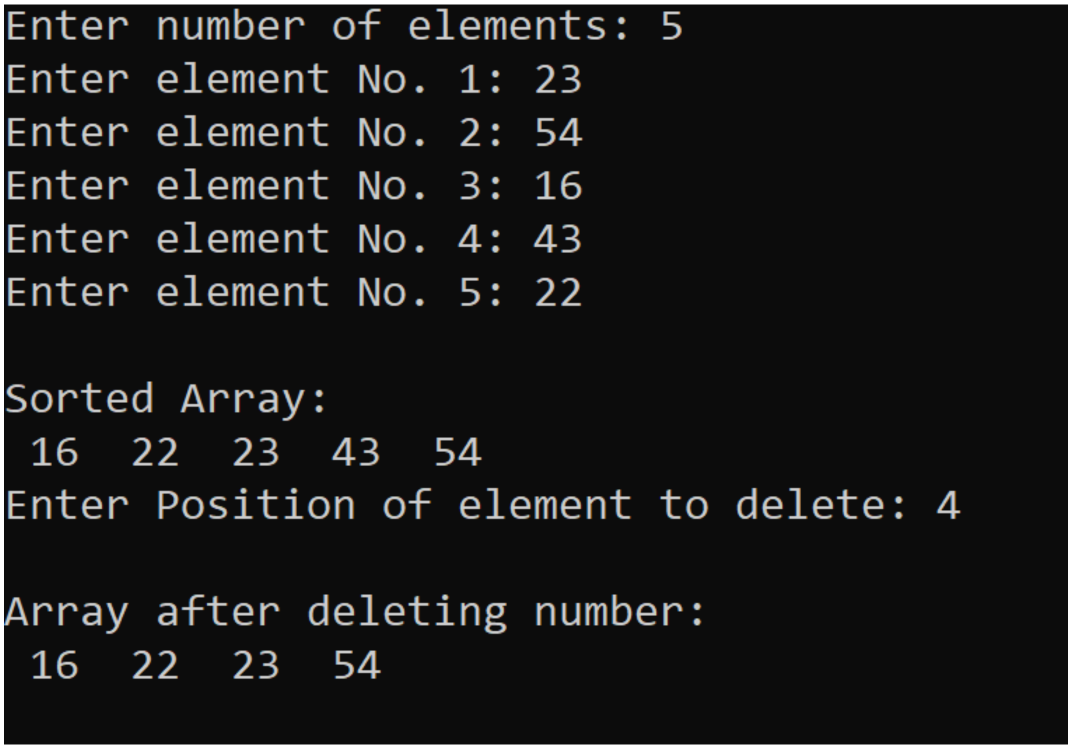
Program:

```
#include<stdio.h>
#include<conio.h>
int main(){
        int num, pos, i,j,x;
        int arr[100];

        printf("Enter number of elements: ");
        scanf("%d",&num);

        for(i=0;i<num;i++){
                printf("Enter element No. %d: ",i+1);
                scanf("%d",&arr[i]);
        }

//Sorting array
        for(i = 0; i<num; i++){
                for(j = i+1; j<num;j++){
                        if(arr[i]>arr[j]){
                                x = arr[i];
                                arr[i] = arr[j];
                                arr[j] = x;
                        }
                }
        }

        printf("\nSorted Array: \n");
        for(i=0;i<num;i++){
                printf(" %d ",arr[i]);
        }

//Deleting element at a position
           printf("\nEnter Position of element to delete: ");
           scanf("%d",&pos);

if (pos>=num){
              printf("\nElement does not exist");
           }
else{
              for (i=pos-1; i<num-1; i++){
              arr[i]=arr[i+1];
              }

   printf("\nArray after deleting number: \n");
   // display the final array
              for (i = 0; i< num - 1; i++)
              {
                 printf (" %d ", arr[i]);
              }

   }
getch();
        return 0;
}
```

Output:

```
  Enter number of elements: 5
Enter element No. 1: 23
Enter element No. 2: 54
Enter element No. 3: 16
Enter element No. 4: 43
Enter element No. 5: 22

Sorted Array:
 16  22  23  43  54
Enter Position of element to delete: 4

Array after deleting number:
 16  22  23  54
```

In Editor:



```
Enter number of elements: 5
Enter element No. 1: 23
Enter element No. 2: 54
Enter element No. 3: 16
Enter element No. 4: 43
Enter element No. 5: 22

Sorted Array:
 16   22   23   43   54
Enter Position of element to delete: 4

Array after deleting number:
 16   22   23   54
```

Q4. Write a program to read and display a 3*3 matrix.

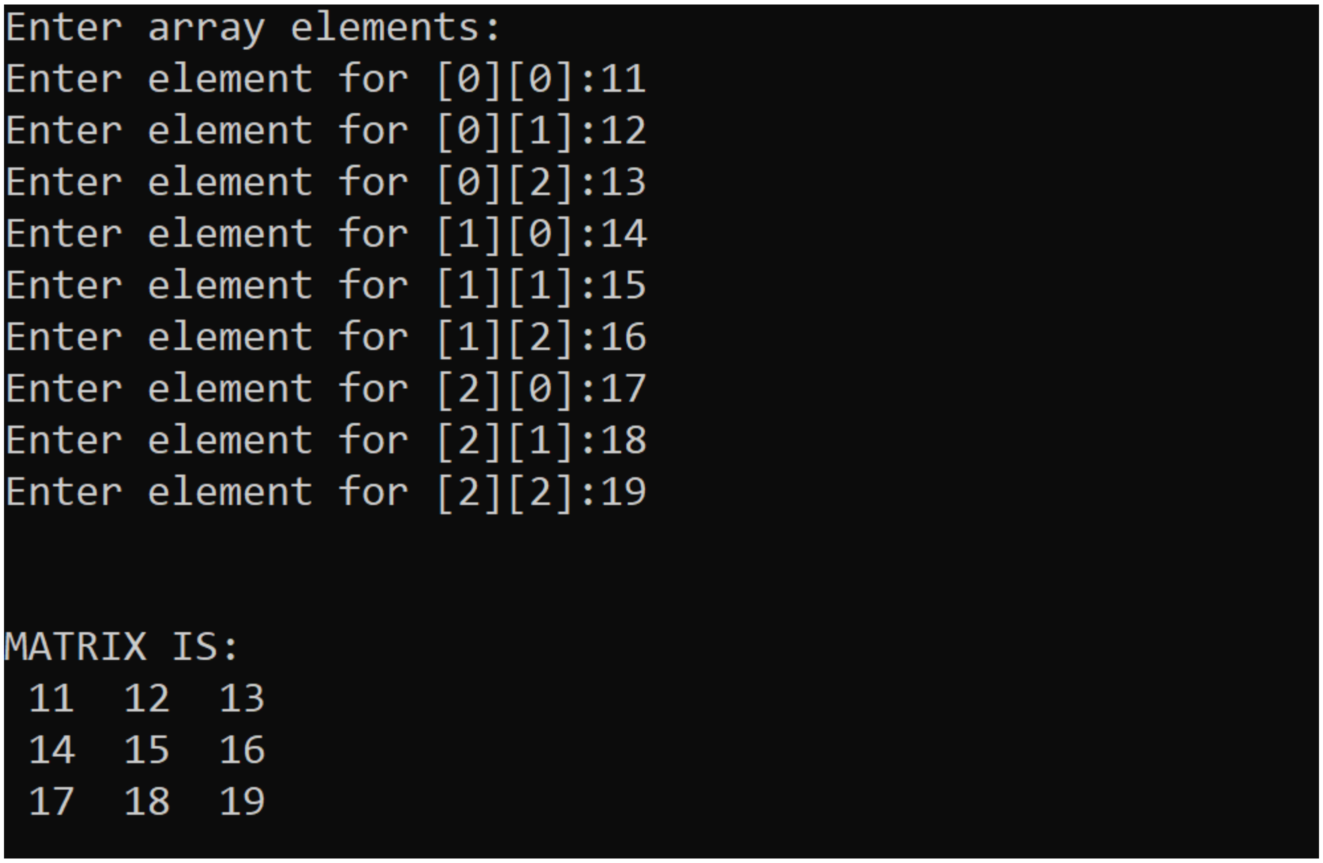
Program:

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

int main(){
        int i,j;
        int matrix[3][3];

        printf("Enter array elements:\n");
        for(i=0;i<3;i++){
            for(j=0;j<3;j++){
                printf("Enter element for [%d][%d]:",i,j);
                scanf("%d",&matrix[i][j]);
            }
        }

        printf("\n\nMATRIX IS:\n");

        for(i=0;i<3;i++){
            for(j=0;j<3;j++){
                printf(" %d ",matrix[i][j]);
            }
            printf("\n");
        }
getch();
return 0;
}
```

Output:

```
Enter array elements:
Enter element for [0][0]:11
Enter element for [0][1]:12
Enter element for [0][2]:13
Enter element for [1][0]:14
Enter element for [1][1]:15
Enter element for [1][2]:16
Enter element for [2][0]:17
Enter element for [2][1]:18
Enter element for [2][2]:19


MATRIX IS:
 11  12  13
 14  15  16
 17  18  19
```

In Editor:



```
Enter array elements:
Enter element for [0][0]:11
Enter element for [0][1]:12
Enter element for [0][2]:13
Enter element for [1][0]:14
Enter element for [1][1]:15
Enter element for [1][2]:16
Enter element for [2][0]:17
Enter element for [2][1]:18
Enter element for [2][2]:19


MATRIX IS:
 11  12  13
 14  15  16
 17  18  19
```

Q5. Write a program to find transpose of a given matrix.

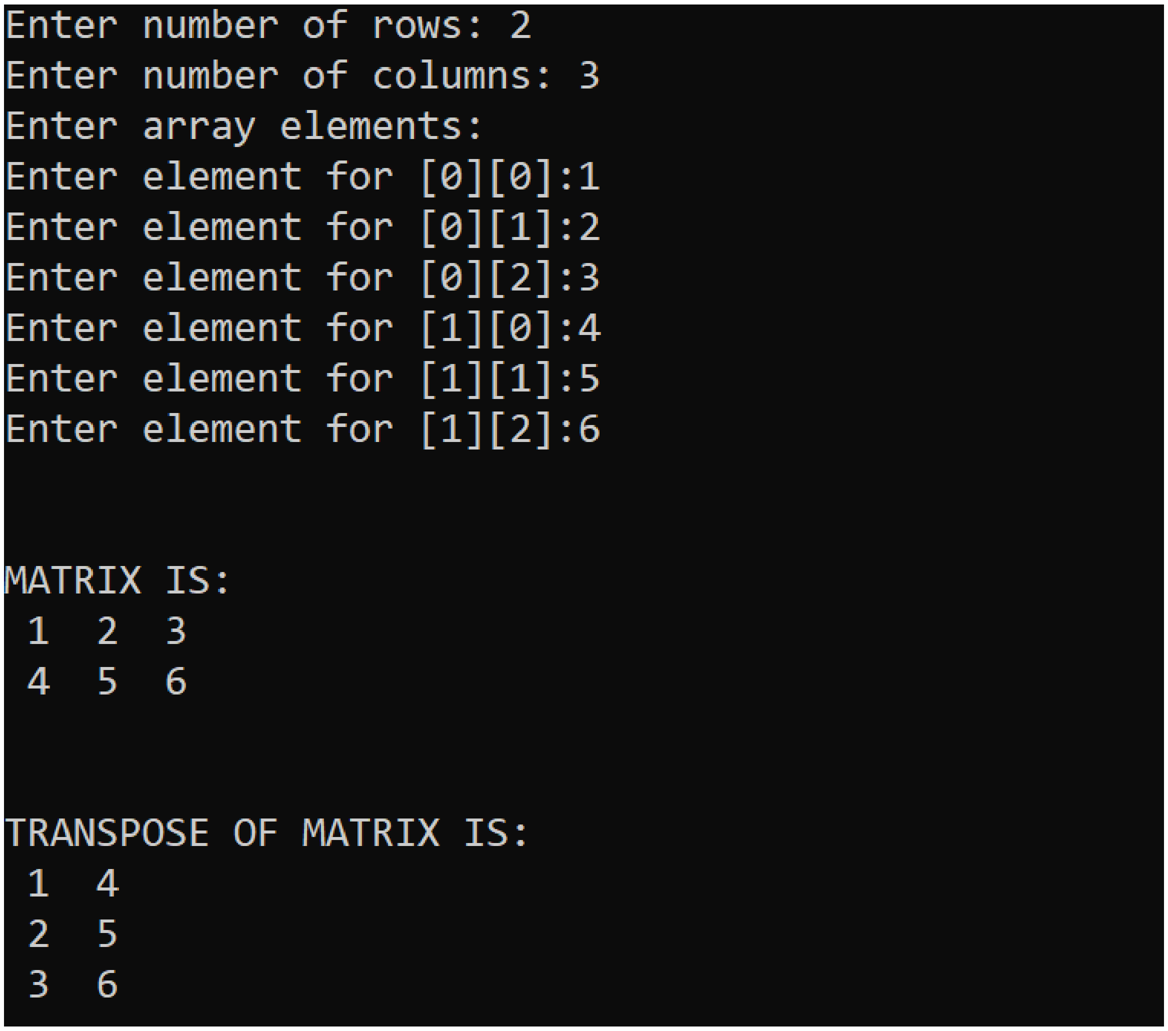
Program:

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

int main(){
   int i,j,row,col;
   int matrix[50][50];

   printf("Enter number of rows: ");
   scanf("%d",&row);

   printf("Enter number of columns: ");
   scanf("%d",&col);

   printf("Enter matrix elements:\n");
            for(i=0;i<row;i++){
               for(j=0;j<col;j++){
                  printf("Enter element for [%d][%d]:",i,j);
                  scanf("%d",&matrix[i][j]);
               }
            }

   printf("\n\nMATRIX IS:\n");

            for(i=0;i<row;i++){
               for(j=0;j<col;j++){
                  printf(" %d ",matrix[i][j]);
               }
               printf("\n");
            }


   printf("\n\nTRANSPOSE OF MATRIX IS:\n");
            for(i=0;i<col;i++){
               for(j=0;j<row;j++){
                  printf(" %d ",matrix[j][i]);
               }
               printf("\n");
            }

getch();
return 0;
}
```

Output:

```
Enter number of rows: 2

Enter number of columns: 3
Enter array elements:
Enter element for [0][0]:1
Enter element for [0][1]:2
Enter element for [0][2]:3
Enter element for [1][0]:4
Enter element for [1][1]:5
Enter element for [1][2]:6


MATRIX IS:
 1  2  3
 4  5  6


TRANSPOSE OF MATRIX IS:
 1  4
 2  5
 3  6
```

In Editor:



```
Enter number of rows: 2
Enter number of columns: 3
Enter array elements:
Enter element for [0][0]:1
Enter element for [0][1]:2
Enter element for [0][2]:3
Enter element for [1][0]:4
Enter element for [1][1]:5
Enter element for [1][2]:6


MATRIX IS:
 1  2  3
 4  5  6


TRANSPOSE OF MATRIX IS:
 1  4
 2  5
 3  6
```

Q6. Write a program to perform transpose of a sparse matrix.

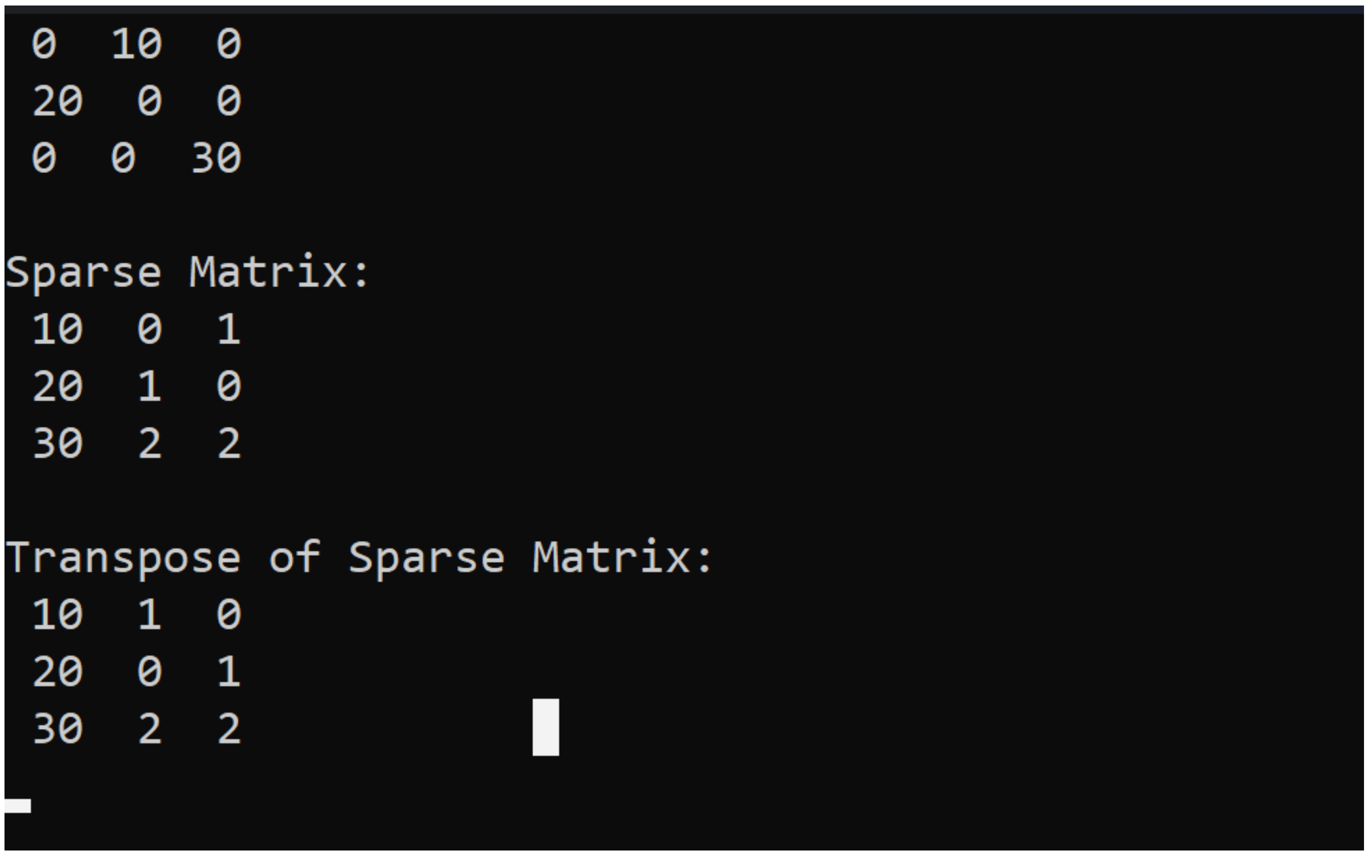
Program:

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

int main(){
        int arr[3][3]={{0,10,0},{20,0,0},{0,0,30}};
        int i,j, count=0,x=1,y,z;
        int sparse[10][3];
        int transpose[10][3];

   for(i=0;i<3;i++){
      for(j=0;j<3;j++){
         printf(" %d ",arr[i][j]);
      }
      printf("\n");
   }
                  for(i=0;i<3;i++){
                       for(j=0;j<3;j++){
                         if(arr[i][j]!=0){
                             sparse[x][0]=arr[i][j];
                             sparse[x][1]=i;
                             sparse[x][2]=j;
                             x++;
                          }
                       }
                    }
   printf("\nSparse Matrix:\n");
                    for(y=1;y<x;y++){
                       for(z=0; z<3; z++){
                          printf(" %d ",sparse[y][z]);
                       }
                       printf("\n");
                    }

                   printf("\nTranspose of Sparse Matrix:\n");
                     for (y=0; y<x; y++){
                       for(z=0; z<3; z++){
                          transpose[y][0]=sparse[y][0];
                          transpose[y][1]=sparse[y][2];
                          transpose[y][2]=sparse[y][1];
                       }
                    }

   for (y=1; y<x; y++){
      for(z=0; z<3; z++){
         printf(" %d ",transpose[y][z]);
      }

      printf("\n");
   }
getch();
return 0;
}
```

Output:

```
0  10  0
 20  0  0
 0  0  30

Sparse Matrix:
 10  0  1
 20  1  0
 30  2  2

Transpose of Sparse Matrix:
 10  1  0
 20  0  1
 30  2  2
```

In Editor:



```
 0  10  0
 20  0  0
 0  0  30

Sparse Matrix:
 10  0  1
 20  1  0
 30  2  2

Transpose of Sparse Matrix:
 10  1  0
 20  0  1
 30  2  2
```

Q7. Write a program to perform addition of two sparse matrices.

Program:

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

int main(){

        int arr1[50][50];
        int arr2[50][50];
        int arr3[50][50];
        int i,j, count=0,x=1,y,z, row,col;
        int sparse1[50][3];
        int sparse2[50][3];
        int sparse3[50][3];

   printf("Enter number of rows: ");
   scanf("%d",&row);

   printf("Enter number of columns: ");
   scanf("%d",&col);

   printf("\nEnter elements for Matrix 1:\n");
      for(i=0;i<row;i++){
         for(j=0;j<col;j++){
            printf("Enter element for [%d][%d]:",i,j);
            scanf("%d",&arr1[i][j]);
         }
      }

   printf("\nEnter elements for Matrix 2:\n");
      for(i=0;i<row;i++){
         for(j=0;j<col;j++){
            printf("Enter element for [%d][%d]:",i,j);
            scanf("%d",&arr2[i][j]);
         }
      }


   printf("\nMatrix 1:\n");
      for(i=0;i<row;i++){
         for(j=0;j<col;j++){
            printf(" %d ",arr1[i][j]);
         }
         printf("\n");
      }

   printf("\nMatrix 2:\n");
      for(i=0;i<row;i++){
         for(j=0;j<col;j++){
            printf(" %d ",arr2[i][j]);
         }
         printf("\n");
      }
```

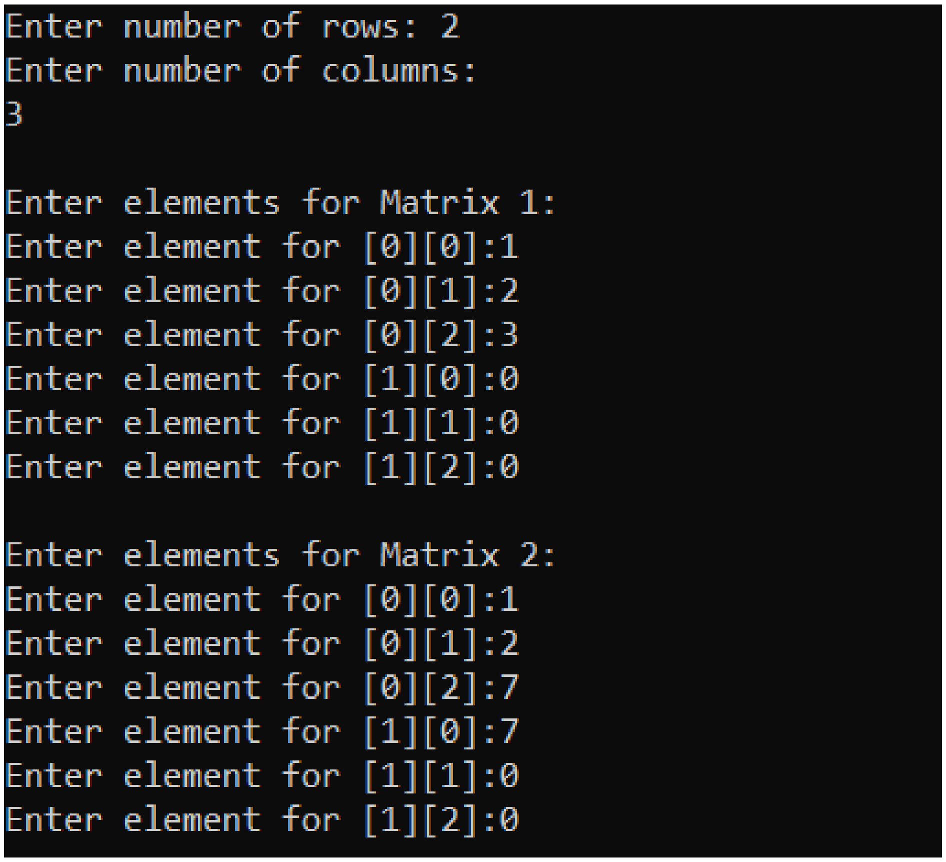
```
//ADDITION OF MATRICES
for(i=0;i<row;i++){
    for (j=0;j<col;j++){
        arr3[i][j] = arr1[i][j] + arr2[i][j];
    }
}

printf("\nMatrix 3:\n");
    for(i=0;i<row;i++){
        for(j=0;j<col;j++){
            printf(" %d ",arr3[i][j]);
        }
        printf("\n");
    }


for(i=0;i<row;i++){
    for(j=0;j<col;j++){
       if(arr1[i][j]!=0){
            sparse1[x][0]=arr1[i][j];
            sparse1[x][1]=i;
            sparse1[x][2]=j;
            x++;
        }
    }
 }

printf("\nSparse Matrix 1:\n");
   for(y=1;y<x;y++){
        for(z=0; z<3; z++){
            printf(" %d ",sparse1[y][z]);
        }
        printf("\n");
    }


 x=1;
 for(i=0;i<row;i++){
    for(j=0;j<col;j++){
       if(arr2[i][j]!=0){
            sparse2[x][0]=arr2[i][j];
            sparse2[x][1]=i;
            sparse2[x][2]=j;
            x++;
        }
    }
 }

printf("\nSparse Matrix 2:\n");
    for(y=1;y<x;y++){
        for(z=0; z<3; z++){
            printf(" %d ",sparse2[y][z]);
        }
        printf("\n");
    }
    x=1;
    for(i=0;i<row;i++){
        for(j=0;j<col;j++){
           if(arr3[i][j]!=0){
                sparse3[x][0]=arr3[i][j];
                sparse3[x][1]=i;
                sparse3[x][2]=j;
                x++;
            }
        }
    }

    printf("\nAddition of Sparse Matrix :\n");
       for(y=1;y<x;y++){
            for(z=0; z<3; z++){
                printf(" %d ",sparse3[y][z]);
            }
            printf("\n");
        }

getch();
return 0;
}
```

Output:

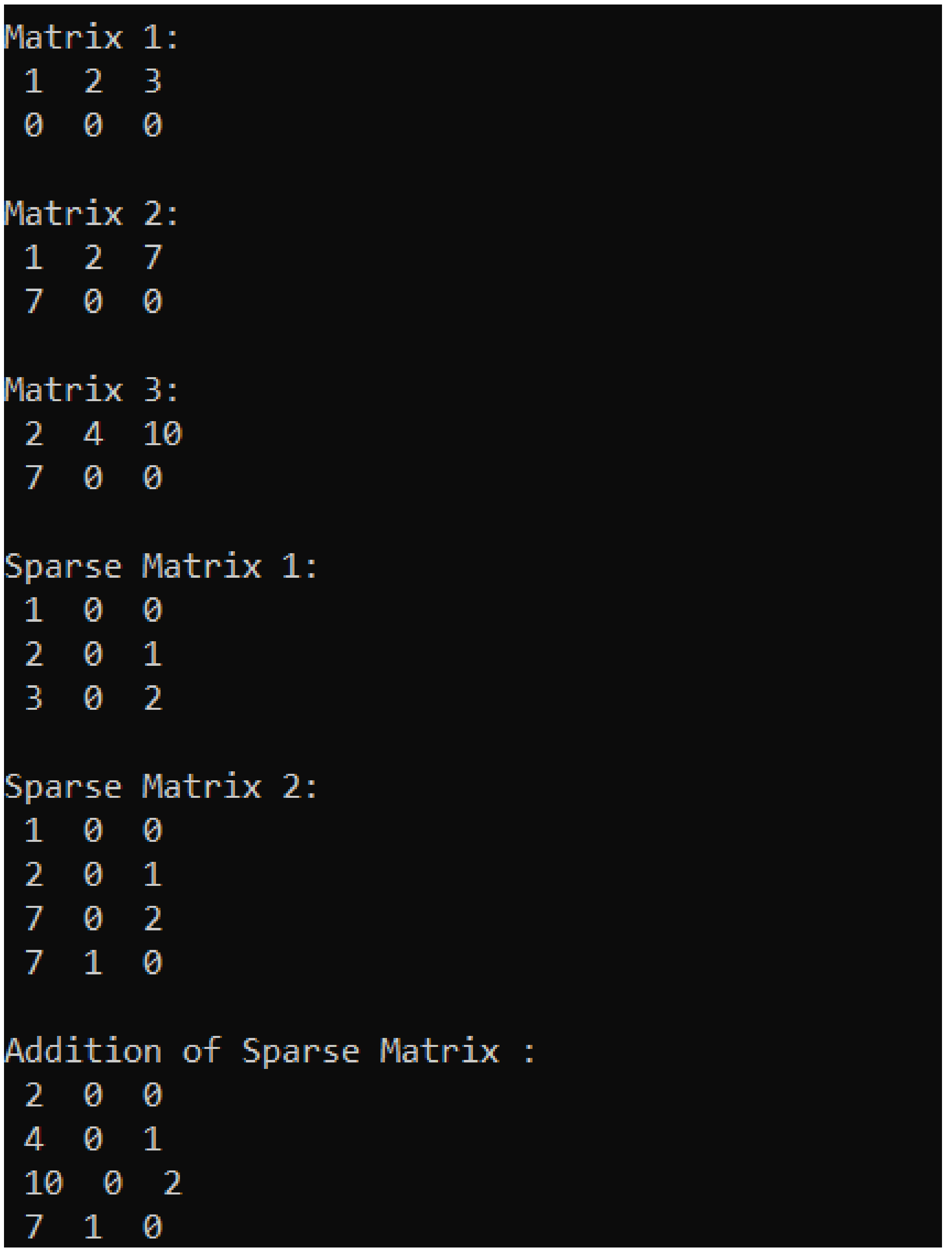
```
Enter number of rows: 2
Enter number of columns: 3

Enter elements for Matrix 1:
Enter element for [0][0]:1
Enter element for [0][1]:2
Enter element for [0][2]:3
Enter element for [1][0]:0
Enter element for [1][1]:0
Enter element for [1][2]:0

Enter elements for Matrix 2:
Enter element for [0][0]:1
Enter element for [0][1]:2
Enter element for [0][2]:7
Enter element for [1][0]:7
Enter element for [1][1]:0
Enter element for [1][2]:0

Matrix 1:
 1  2  3
 0  0  0

Matrix 2:
 1  2  7
 7  0  0
```

```
Matrix 3:
 2  4  10
 7  0  0

Sparse Matrix 1:
 1  0  0
 2  0  1
 3  0  2

Sparse Matrix 2:
 1  0  0
 2  0  1
 7  0  2
 7  1  0

Addition of Sparse Matrix :
 2  0  0
 4  0  1
 10  0  2
 7  1  0
```

In Editor:



```
Enter number of rows: 2
Enter number of columns:
3

Enter elements for Matrix 1:
Enter element for [0][0]:1
Enter element for [0][1]:2
Enter element for [0][2]:3
Enter element for [1][0]:0
Enter element for [1][1]:0
Enter element for [1][2]:0

Enter elements for Matrix 2:
Enter element for [0][0]:1
Enter element for [0][1]:2
Enter element for [0][2]:7
Enter element for [1][0]:7
Enter element for [1][1]:0
Enter element for [1][2]:0
```


```
Matrix 1:
 1  2  3
 0  0  0

Matrix 2:
 1  2  7
 7  0  0

Matrix 3:
 2  4  10
 7  0  0

Sparse Matrix 1:
 1  0  0
 2  0  1
 3  0  2

Sparse Matrix 2:
 1  0  0
 2  0  1
 7  0  2
 7  1  0

Addition of Sparse Matrix :
 2  0  0
 4  0  1
 10  0  2
 7  1  0
```

Q8. Write a program to perform addition of a two polynomial expressions using arrays.

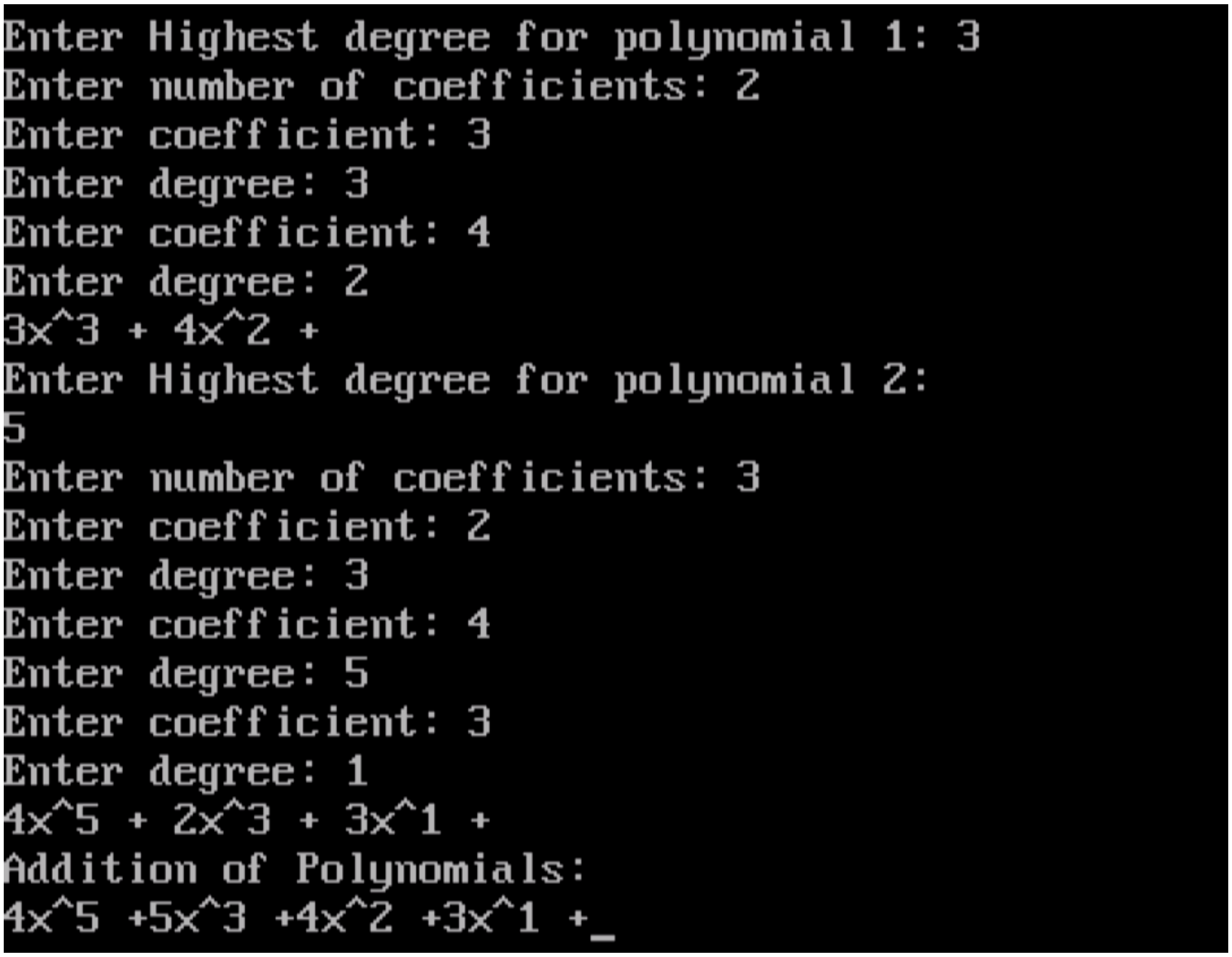
Program:

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

int main(){
    int poly1[50], poly2[50], polyadd[50];
    int coeffiecient, hdegree1, hdegree2, coef,deg;
    int i,j;

    //For  Polynomial 1
    printf("Enter Highest degree for polynomial 1: ");
    scanf("%d",&hdegree1);

    printf("Enter number of coefficients: ");
    scanf("%d",&coeffiecient);

    for (j=0;j<hdegree1+1;j++){
        poly1[j]=0;
    }

    for (i=0; i<coeffiecient; i++){
        printf("Enter coefficient: ");
        scanf("%d",&coef);
        printf("Enter degree: ");
        scanf("%d",&deg);

        poly1[deg]= coef;
    }

    for (i=hdegree1;i>=0;i--){
        if(poly1[i]!=0){
            printf("%dx^%d + ", poly1[i],i);
        }
    }

    //For  Polynomial 2
    printf("\nEnter Highest degree for polynomial 2: ");
    scanf("%d",&hdegree2);

    printf("Enter number of coefficients: ");
    scanf("%d",&coeffiecient);

    for (j=0;j<hdegree2+1;j++){
        poly2[j]=0;
    }

    for (i=0; i<coeffiecient; i++){
        printf("Enter coefficient: ");
        scanf("%d",&coef);
        printf("Enter degree: ");
        scanf("%d",&deg);

        poly2[deg]= coef;
    }
    for (i=hdegree2;i>=0;i--){
        if(poly2[i]!=0){
            printf("%dx^%d + ", poly2[i],i);
        }
    }
    if (hdegree1>hdegree2){
        deg = hdegree1;
    }else if(hdegree2>hdegree1){
        deg = hdegree2;
    }else{
        deg = hdegree1;
    }

    for (j=0;j<=deg;j++){
        polyadd[j]=0;
    }

    for (i=0; i<=deg; i++){
        polyadd[i] = poly1[i] + poly2[i];
    }


    printf("\nAddition of Polynomials: \n");

    for (i=deg+1; i>=0 ; i--){
        if(polyadd[i]!=0){
            printf("%dx^%d +", polyadd[i],i);
        }
    }

getch();
return 0;
}
```

Output:

```
Enter Highest degree for polynomial 1: 3
Enter number of coefficients: 2
Enter coefficient: 3
Enter degree: 3
Enter coefficient: 4
Enter degree: 2
3x^3 + 4x^2 +
Enter Highest degree for polynomial 2: 5
Enter number of coefficients: 3
Enter coefficient: 2
Enter degree: 3
Enter coefficient: 4
Enter degree: 5
Enter coefficient: 3
Enter degree: 1
4x^5 + 2x^3 + 3x^1 +
Addition of Polynomials:
4x^5 + 5x^3 + 4x^2 + 3x^1 +
```

In Editor:



```
Enter Highest degree for polynomial 1: 3
Enter number of coefficients: 2
Enter coefficient: 3
Enter degree: 3
Enter coefficient: 4
Enter degree: 2
3x^3 + 4x^2 +
Enter Highest degree for polynomial 2:
5
Enter number of coefficients: 3
Enter coefficient: 2
Enter degree: 3
Enter coefficient: 4
Enter degree: 5
Enter coefficient: 3
Enter degree: 1
4x^5 + 2x^3 + 3x^1 +
Addition of Polynomials:
4x^5 +5x^3 +4x^2 +3x^1 +_
```

Progressive Education Society's

Modern College of Engineering, Pune

MCA Department

A.Y.2022-23

(310902) Data Structure and Algorithm Laboratory

****

Class: FY-MCA Shift /Div: A Batch: F2 Roll Number: 51043

Name: Vanessa Reetu Prashant More Assignment No: 2 Date of Implementation: 29/12/2022

****

Q1. Write a menu driven program to perform following operations on singly linked list: Create, Insert, Delete, and Display.

Program:

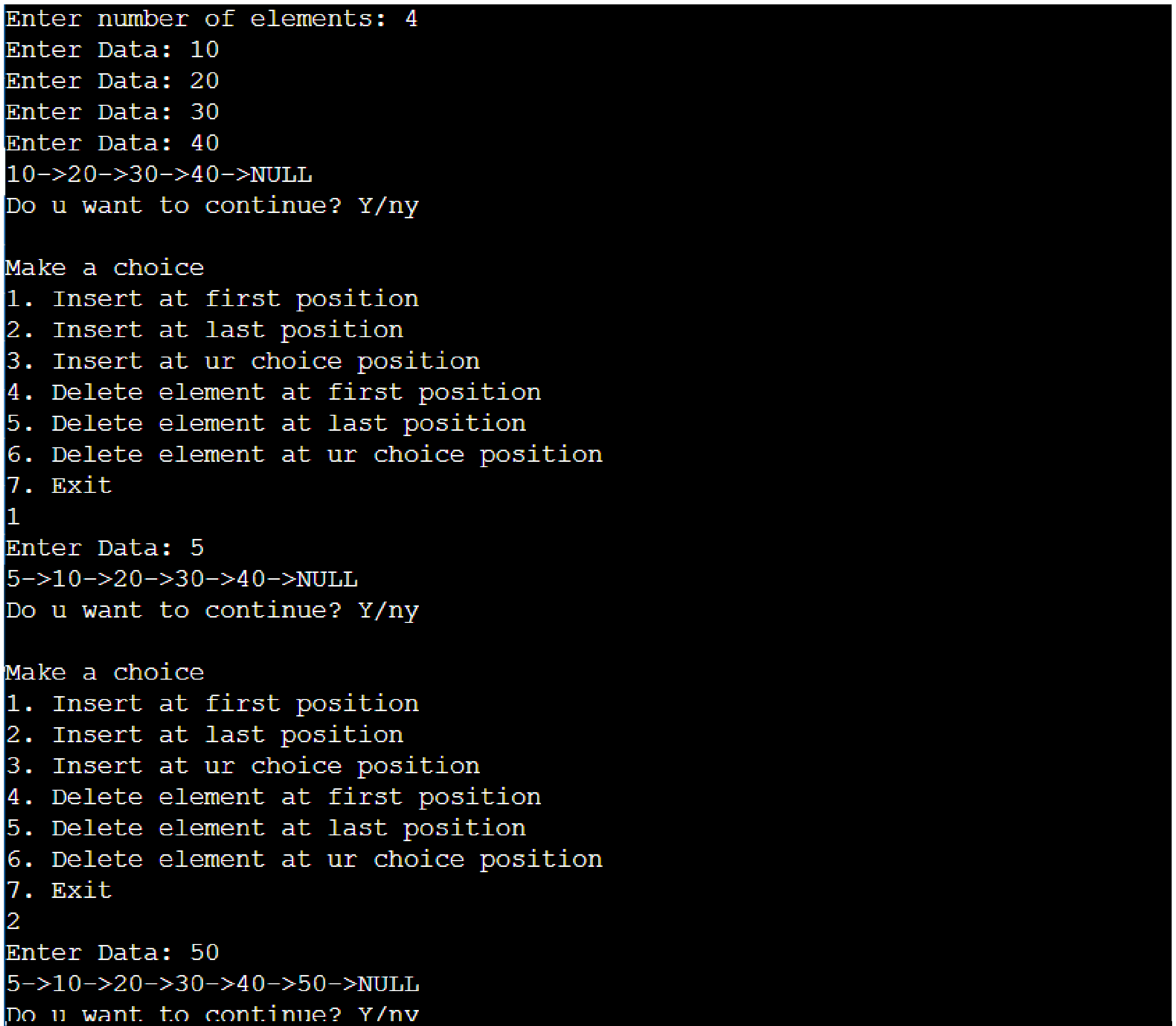
```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

struct Node
{ int data;
 struct Node *next;

}*head=NULL, *end;

void create(){
        struct Node *temp;
        temp = (struct Node *)malloc(sizeof(struct Node));
        printf("Enter Data: ");
        scanf("%d",&temp->data);

        if(head==NULL){
                head=temp;
                end=temp;
        }
        else{
                end->next=temp;
                end=temp;
        }
}

void insert_end(){
        struct Node *temp;
        temp = (struct Node *)malloc(sizeof(struct Node));
        printf("Enter Data: ");
        scanf("%d",&temp->data);
                end->next=temp;
                end=temp;
}

void insert_start(){
        struct Node *temp;
        temp = (struct Node *)malloc(sizeof(struct Node));
        printf("Enter Data: ");
        scanf("%d",&temp->data);
        temp->next = head;
        head =temp;
}

void insert_pos(){
    int pos,i;
    struct Node *t1, *t2, *temp;
    temp = (struct Node *)malloc(sizeof(struct Node));
    printf("Enter Data: ");
    scanf("%d",&temp->data);
    printf("Enter Position: ");
    scanf("%d",&pos);

    t2=head;
    for(i=0;i<pos-1;i++){
        t1=t2;
        t2=t1->next;
    }

    t1->next=temp;
    temp->next=t2;

}

void del_last(){
        struct Node *t1, *t2;
        t2=head;

        while(t2->next!=NULL){
                t1=t2;
                t2=t2->next;
        }

        t1->next = NULL;
        end=t1;
        free(t2);
}

void del_first(){
        struct Node *t1;
        t1=head;
        head = t1->next;
        free(t1);
}

void del_pos(){
        int i, pos;
        struct Node *t1, *t2;
        t2 =head;

        printf("Enter position: ");
        scanf("%d",&pos);
        for(i=0; i<pos-1;i++)
        {       t1=t2;
                t2 = t1->next;
        }

        t1->next=t2->next;
        free(t2);
}

void display(){
        struct Node *temp;
        temp=head;
        while (temp!=NULL){
                printf("%d->",temp->data);
                temp = temp->next;
        }
        printf("NULL");

}

int main(){
        int x,i,c;
        char con;
        //clrscr();
        printf("Enter number of elements: ");
        scanf("%d",&x);

        for (i=0;i<x; i++){
                create();
        }
        display();

        printf("\nDo u want to continue? Y/n");
        scanf("%s", &con);


        while ( con =='y'|| con == 'Y'){
        printf("\nMake a choice\n");
        printf("1. Insert at first position\n");
        printf("2. Insert at last position\n");
        printf("3. Insert at ur choice position\n");
        printf("4. Delete element at first position\n");
        printf("5. Delete element at last position\n");
        printf("6. Delete element at ur choice position\n");
        printf("7. Exit\n");
        scanf("%d",&c);
switch(c){
                case 1: insert_start();
                        display();
                        break;

                case 2: insert_end();
                        display();
                        break;

                case 3: insert_pos();
                        display();
                        break;

                case 4: del_first();
                        display();
                        break;

                case 5: del_last();
                        display();
                        break;

                case 6: del_pos();
                        display();
                        break;

                 case 7: exit(0);
                default: printf("Wrong choice:");
        }

        printf("\nDo u want to continue? Y/n");
        scanf("%s", &con);
        }
        getch();
        return 0;

}
```

Output:

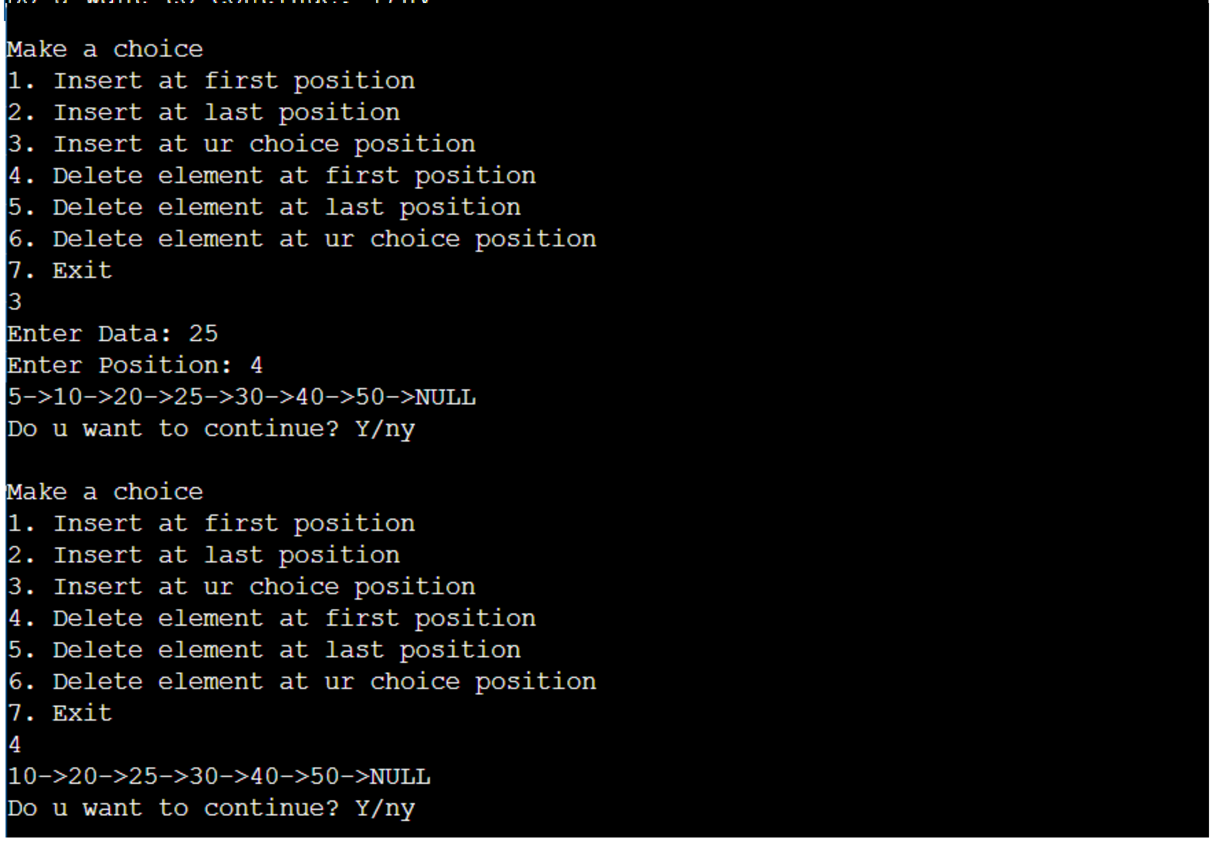
```
Enter number of elements: 4
Enter Data: 10
Enter Data: 20
Enter Data: 30
Enter Data: 40
10->20->30->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
1
Enter Data: 5
5->10->20->30->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
2
Enter Data: 50
5->10->20->30->40->50->NULL
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
3
Enter Data: 25
Enter Position: 4
5->10->20->25->30->40->50->NULL
Do u want to continue? Y/ny
```

```
Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
4
10->20->25->30->40->50->NULL
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
5
10->20->25->30->40->NULL
Do u want to continue? Y/n6_ _y

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
6
Enter position: 4
10->20->25->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
7
```

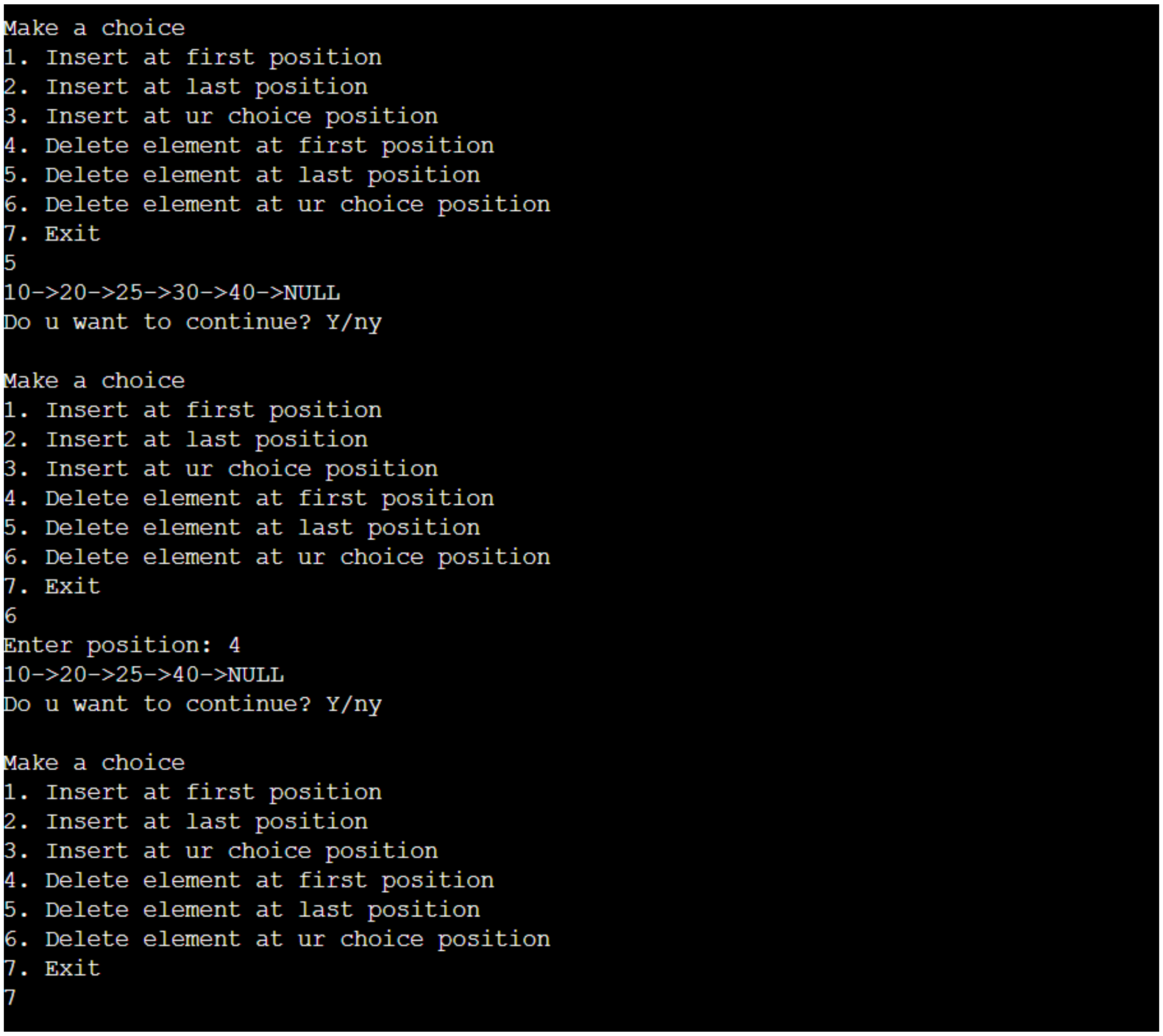
In Editor:



```
Enter number of elements: 4
Enter Data: 10
Enter Data: 20
Enter Data: 30
Enter Data: 40
10->20->30->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
1
Enter Data: 5
5->10->20->30->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
2
Enter Data: 50
5->10->20->30->40->50->NULL
Do u want to continue? Y/ny
```



```
Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
3
Enter Data: 25
Enter Position: 4
5->10->20->25->30->40->50->NULL
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
4
10->20->25->30->40->50->NULL
Do u want to continue? Y/ny
```



```
Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
5
10->20->25->30->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
6
Enter position: 4
10->20->25->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
7
```

---

2. Write a menu driven program to perform following operations on singly linked list: Create, reverse, search, count and Display.

Program:

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

struct Node
{ int data;
 struct Node *next;

}*head=NULL, *end;

void create(){
        struct Node *temp;
        temp = (struct Node *)malloc(sizeof(struct Node));
        printf("Enter Data: ");
        scanf("%d",&temp->data);

        if(head==NULL){
                head=temp;
                end=temp;
        }
        else{
                end->next=temp;
                end=temp;
        }
}

void search(){
   struct Node *t;
   int n, c=1;

   printf("Enter node to check: ");
   scanf("%d",&n);
   t=head;
   while (t!=NULL){

      if (t->data== n){

         printf("Node is present at position: %d\n ",c);
      }
      t=t->next;
      c++;
   }
}
```

```
void count(){

    struct Node *t;
    int c=0;
    t=head;
    while(t!=NULL){
        t=t->next;
        c++;
    }

    printf("\nCount of number of nodes = %d ",c);

}

void reverse(){

    struct Node *next, *prev, *cur;
    prev = head;

    cur = head->next;
    head= head->next;
    prev->next=NULL;

    end = prev;

    while(head!=NULL){
        head = head->next;
        cur->next=prev;
        prev = cur;
        cur=head;
    }

    head=prev;

}

void display(){
            struct Node *temp;
            temp=head;
            while (temp!=NULL){
                        printf("%d->",temp->data);
                        temp = temp->next;
            }
            printf("NULL");

}
```

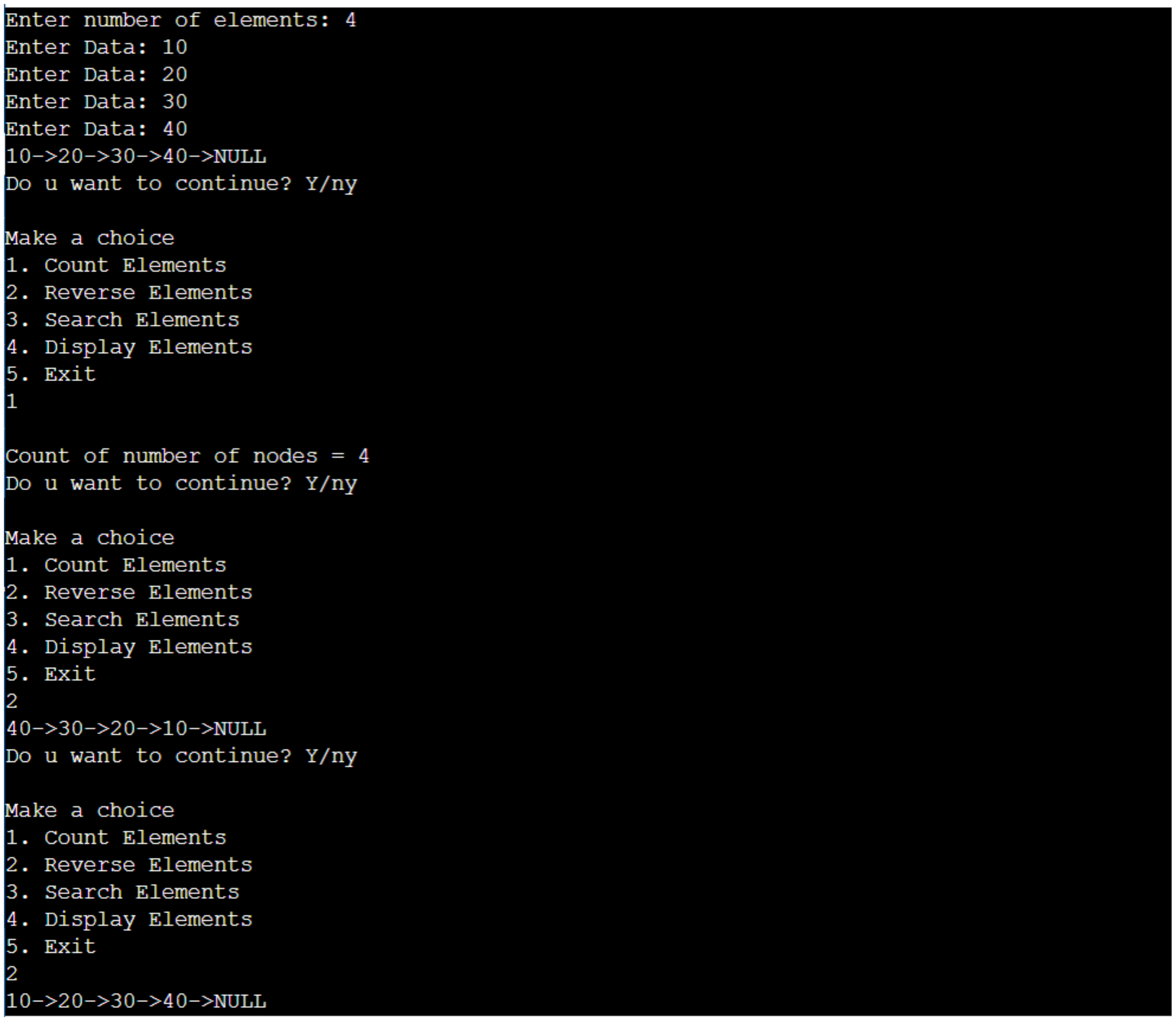
```
int main(){
        int x,i,c;
        char con;
        //clrscr();
        printf("Enter number of elements: ");
        scanf("%d",&x);

        for (i=0;i<x; i++){
                create();
        }
        display();

        printf("\nDo u want to continue? Y/n");
        scanf("%s", &con);


        while ( con =='y'|| con == 'Y'){
        printf("\nMake a choice\n");
        printf("1. Count Elements\n");
        printf("2. Reverse Elements\n");
        printf("3. Search Elements\n");
        printf("4. Display Elements\n");
        printf("5. Exit\n");
        scanf("%d",&c);

        switch(c){
                case 1: count();
                        //display();
                        break;

                case 2:reverse();
                        display();
                        break;

                case 3: search();
                        display();
                        break;

                case 4: display();
                        break;

                case 5: exit(0);
                        break;

                default: printf("Wrong choice:");
        }

        printf("\nDo u want to continue? Y/n");
        scanf("%s", &con);
        }
        //getch();
        return 0;

}
```

Output:

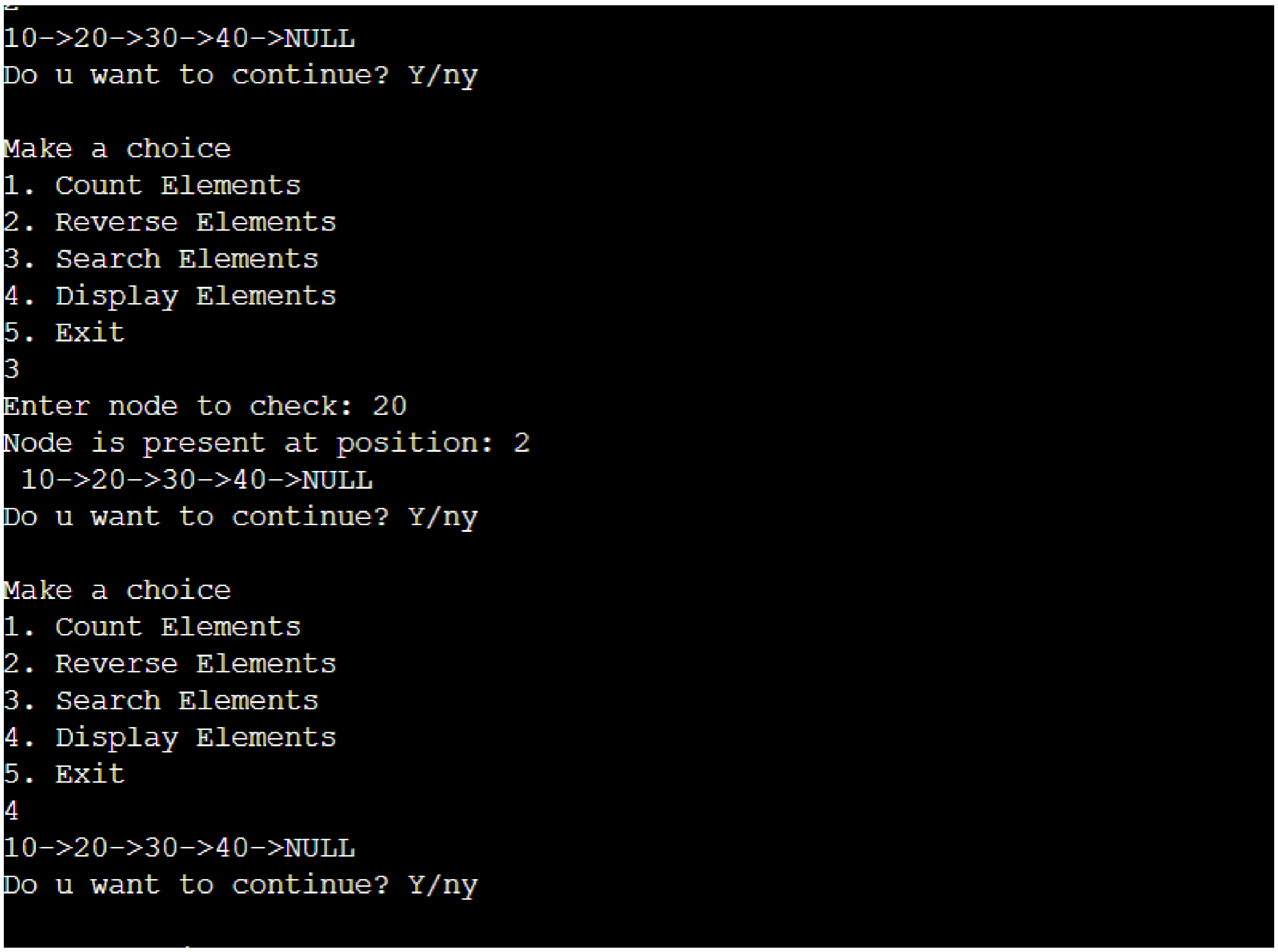
```
Enter number of elements: 4
Enter Data: 10
Enter Data: 20
Enter Data: 30
Enter Data: 40
10->20->30->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
1

Count of number of nodes = 4
Do u want to continue? Y/ny

Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
2
40->30->20->10->NULL
Do u want to continue? Y/ny

Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
2
10->20->30->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
3
Enter node to check: 20
Node is present at position: 2
 10->20->30->40->NULL
Do u want to continue? Y/ny
```

```
Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
4
10->20->30->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
5
```

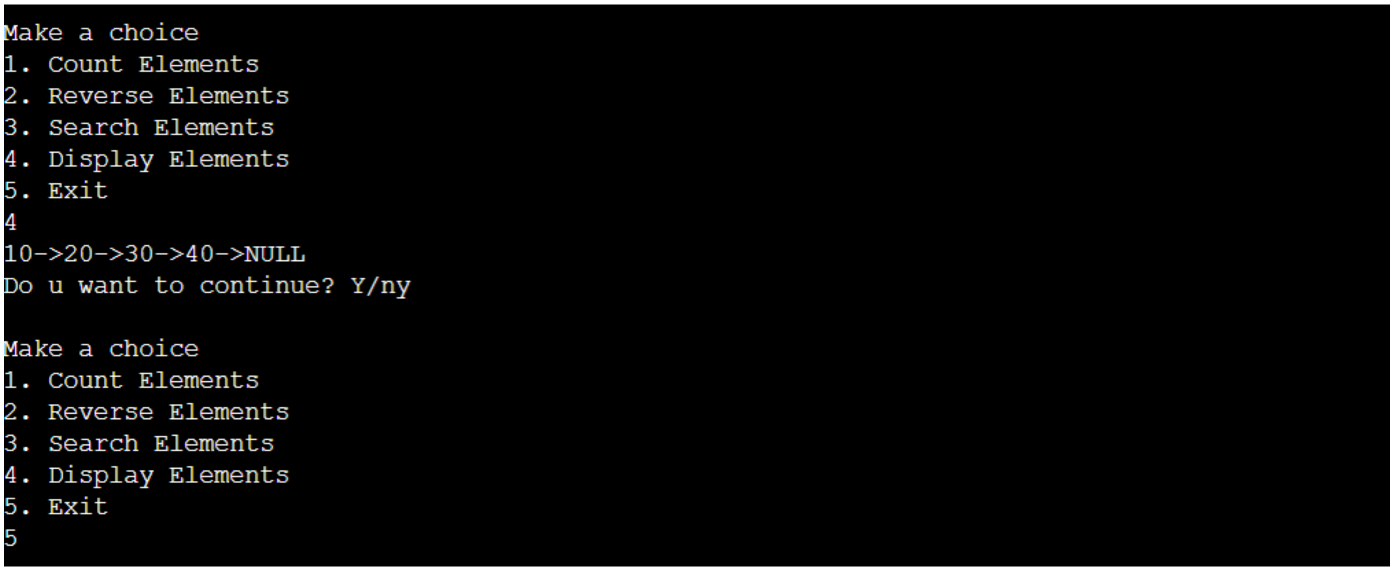
In Editor:



```
Enter number of elements: 4
Enter Data: 10
Enter Data: 20
Enter Data: 30
Enter Data: 40
10->20->30->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
1

Count of number of nodes = 4
Do u want to continue? Y/ny

Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
2
40->30->20->10->NULL
Do u want to continue? Y/ny

Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
2
10->20->30->40->NULL
```



```
10->20->30->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
3
Enter node to check: 20
Node is present at position: 2
 10->20->30->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
4
10->20->30->40->NULL
Do u want to continue? Y/ny
```



```
Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
4
10->20->30->40->NULL
Do u want to continue? Y/ny

Make a choice
1. Count Elements
2. Reverse Elements
3. Search Elements
4. Display Elements
5. Exit
5
```

Q3. Write a menu driven program to perform operations on doubly linked list: Create, Insert, Delete, and Display

Program:

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

struct Node{
        int data;
        struct Node *prev;
        struct Node *next;
}*head=NULL, *end=NULL;

void create(){
        struct Node *temp;
        temp= (struct Node *)malloc(sizeof(struct Node));

        printf("Enter the data: ");
        scanf("%d",&temp->data);

        temp->next=NULL;
        temp->prev=NULL;

        if(head==NULL){
                head=temp;
                end=temp;
        }
        else{
                end->next=temp;
                end->prev=end;
                end=temp;
        }
}

void insert_first(){
        struct Node *temp;
        printf("Inserting element at 1st position:\n");

        temp=(struct Node *)malloc(sizeof(struct Node));

        printf("Enter data: ");
        scanf("%d",&temp->data);

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

```
void insert_end(){
        struct Node *temp;
        printf("Inserting element at end position:\n");

        temp=(struct Node *)malloc(sizeof(struct Node));

        printf("Enter data: ");
        scanf("%d",&temp->data);

        end->next=temp;
        temp->next=NULL;
        temp->prev=NULL;
        temp->prev=end;
        end=temp;
}
void insert_pos(){
        int pos,i;
        struct Node *t1, *t2, *temp;
   temp=(struct Node *)malloc(sizeof(struct Node));
        printf("Inserting element at choice position: \n");
        printf("Enter position: ");
        scanf("%d",&pos);
        printf("Enter data: ");
        scanf("%d",&temp->data);

        t2=head;
        for(i=0; i<pos-1; i++){
                t1=t2;
                t2=t2->next;
        }

   temp->next=t2;
        t1->next=temp;
        t2->prev=temp;
        temp->prev=t1;
}

void delete_end(){
        struct Node *temp, *t;
        /*temp = end;
        end = end->prev;
        end->next=NULL;
        end=NULL;
        free(temp);*/

        temp = head;
        while(temp->next!=NULL){
                t=temp;
                temp = temp->next;
        }
        t->next=NULL;
        free(temp);


}
```

```
void delete_start(){
	struct Node *temp;

	temp=head;
	head=head->next;
	free(temp);
}

void delete_pos(){
	int pos,i;
	struct Node *t1, *temp;

	printf("Enter position to delete:");
	scanf("%d",&pos);

	temp=head;
	for(i=0; i<pos-1; i++){
		t1=temp;
		temp=temp->next;
	}

	//temp=t1->next;
    //	t1->next =t2;
	t1->next=temp->next;
	t1->next->prev=t1;
	free(temp);
}

void display(){
	struct Node *t1, *t2, *temp;
	temp=head;
	//t2=end;
	printf("NULL->");
	while(temp!=NULL){
		printf("%d->",temp->data);
		temp = temp->next;
	}

	printf("NULL\n");

    //	printf("NULL");
}

void main(){

	int x,i,c;
	char con;
	//clrscr();
	printf("Enter number of elements: ");
	scanf("%d",&x);

	for (i=0;i<x; i++){
		create();
	}
```

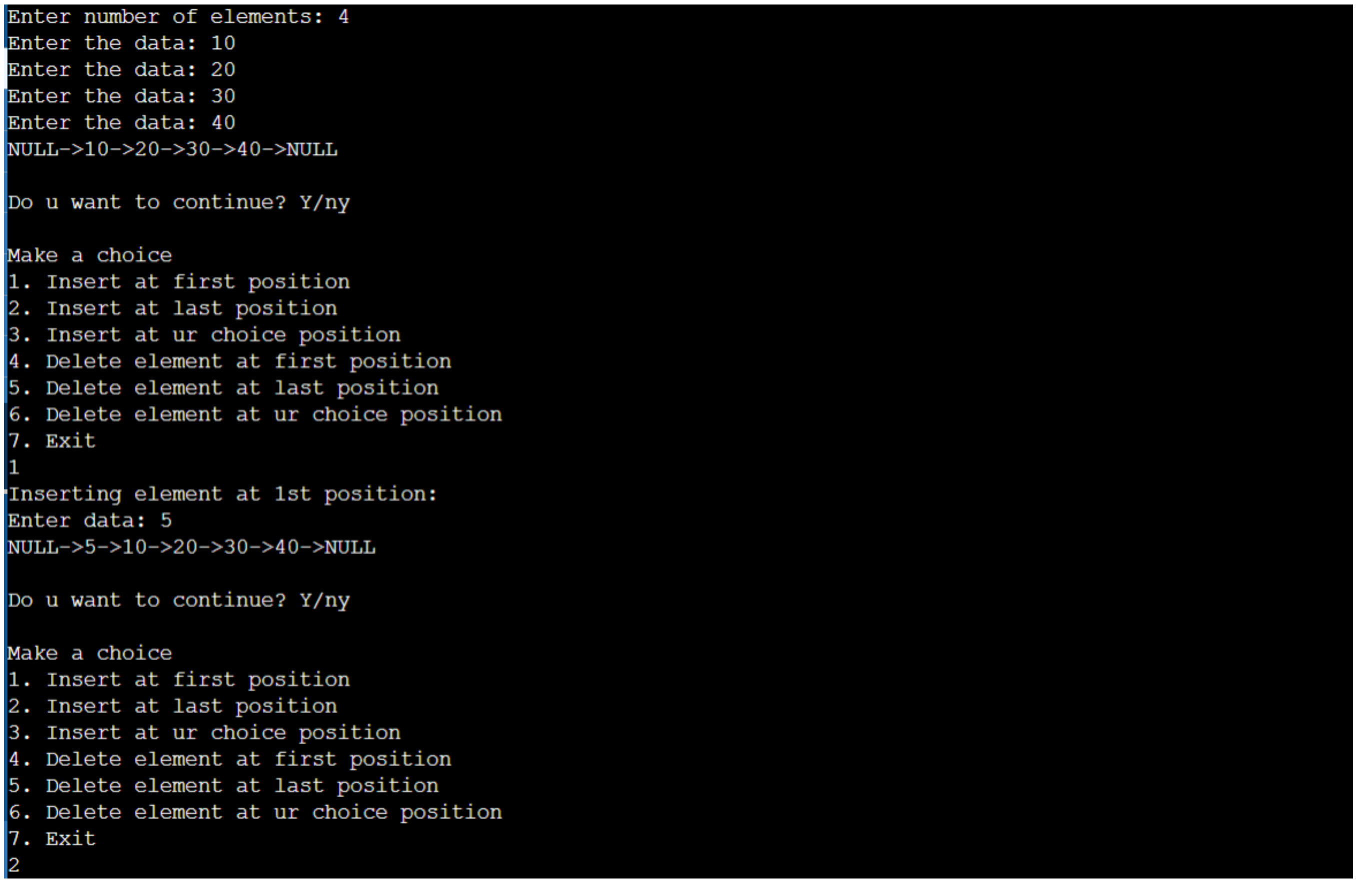
```
        display();

        printf("\nDo u want to continue? Y/n");
        scanf("%s", &con);

        while ( con =='y'|| con == 'Y'){
        printf("\nMake a choice\n");
        printf("1. Insert at first position\n");
        printf("2. Insert at last position\n");
        printf("3. Insert at ur choice position\n");
        printf("4. Delete element at first position\n");
        printf("5. Delete element at last position\n");
        printf("6. Delete element at ur choice position\n");
        printf("7. Exit\n");
        scanf("%d",&c);

        switch(c){
                case 1: insert_first();
                        display();
                        break;

                case 2: insert_end();
                        display();
                        break;

                case 3: insert_pos();
                        display();
                        break;

                case 4: delete_start();
                        display();
                        break;

                case 5: delete_end();
                        display();
                        break;

                case 6: delete_pos();
                        display();
                        break;

           case 7: exit(0);
                default: printf("Wrong choice:");
        }

   printf("\nDo u want to continue? Y/n");
        scanf("%s", &con);
        }

getch();
}
```

Output:

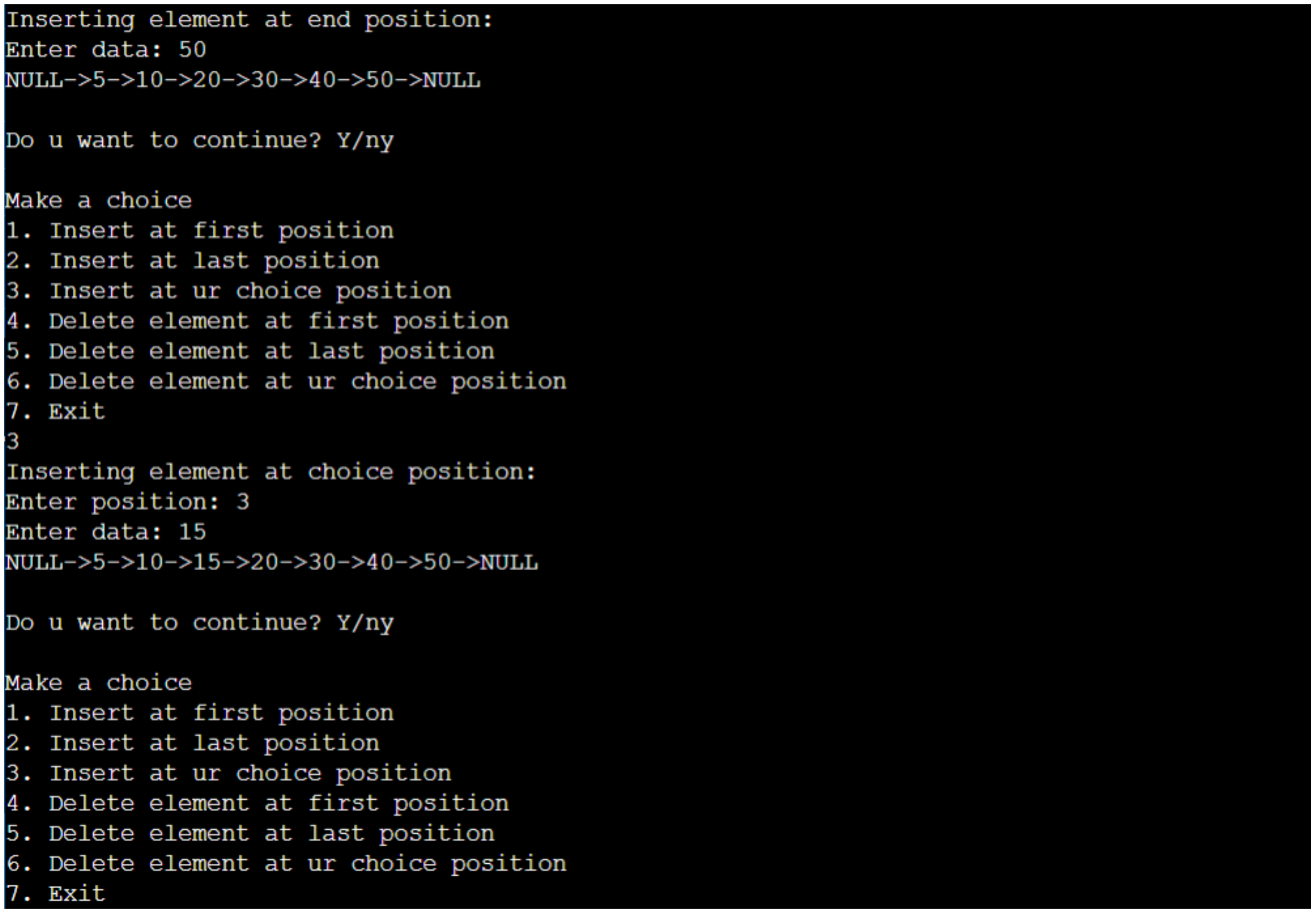
```
Enter number of elements: 4
Enter the data: 10
Enter the data: 20
Enter the data: 30
Enter the data: 40
NULL->10->20->30->40->NULL

Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
1
Inserting element at 1st position:
Enter data: 5
NULL->5->10->20->30->40->NULL

Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
2
Inserting element at end position:
Enter data: 50
NULL->5->10->20->30->40->50->NULL

Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
3
Inserting element at choice position:
Enter position: 3
Enter data: 15
NULL->5->10->15->20->30->40->50->NULL
```

```
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
y_ _4
NULL->10->15->20->30->40->50->NULL

Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
5
NULL->10->15->20->30->40->NULL

Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
6
Enter position to delete:3
NULL->10->15->30->40->NULL

Do u want to continue? Y/nn
```

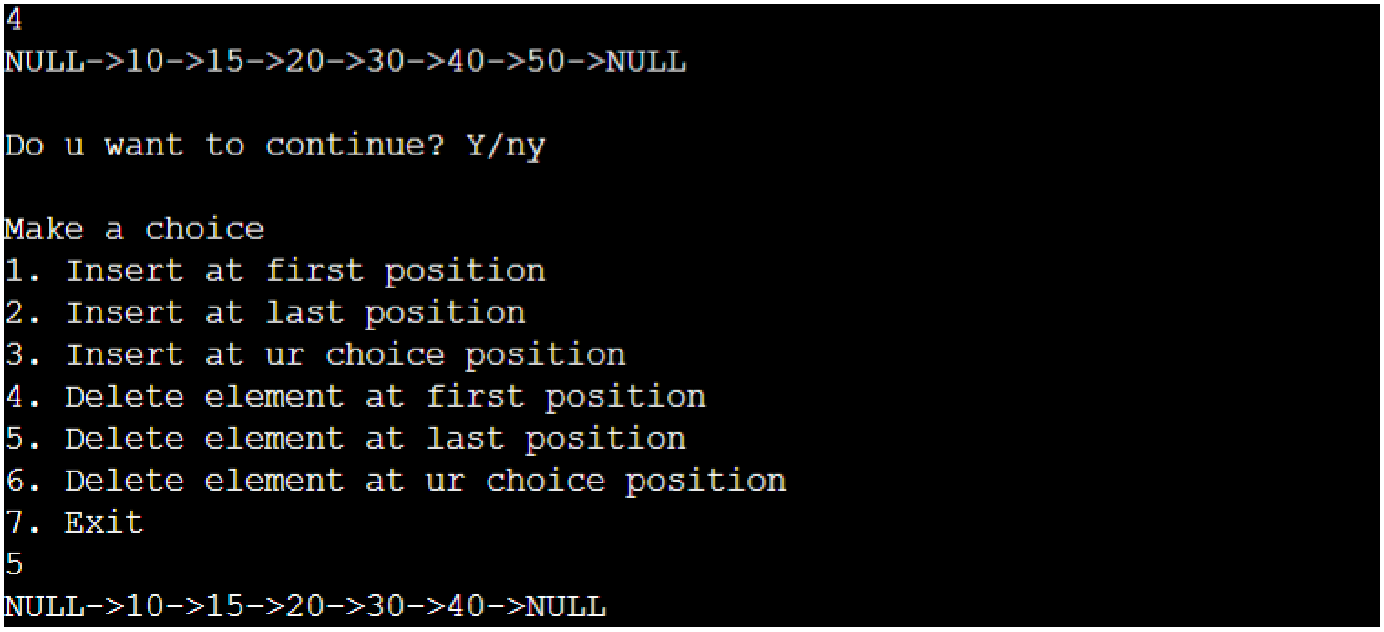
In Editor:



```
Enter number of elements: 4
Enter the data: 10
Enter the data: 20
Enter the data: 30
Enter the data: 40
NULL->10->20->30->40->NULL

Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
1
Inserting element at 1st position:
Enter data: 5
NULL->5->10->20->30->40->NULL

Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
2
```



```
Inserting element at end position:
Enter data: 50
NULL->5->10->20->30->40->50->NULL

Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
3
Inserting element at choice position:
Enter position: 3
Enter data: 15
NULL->5->10->15->20->30->40->50->NULL

Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
```



```
4
NULL->10->15->20->30->40->50->NULL

Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
5
NULL->10->15->20->30->40->NULL
```

Q4. Implement circular linked list and perform operations on it: Create, Insert, Delete, and Display

Program:

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

struct Node
{ int data;
 struct Node *next;

}*head=NULL, *end;

void create(){
        struct Node *temp;
        temp = (struct Node *)malloc(sizeof(struct Node));
        printf("Enter Data: ");
        scanf("%d",&temp->data);

        if(head==NULL){
                head=temp;
                end=temp;
                end->next=head;
        }
        else{
                end->next=temp;
                end=temp;
                end->next = head;
        }
}

void insert_end(){
        struct Node *temp;
        temp = (struct Node *)malloc(sizeof(struct Node));
        printf("Enter Data: ");
        scanf("%d",&temp->data);
                end->next=temp;
                end=temp;
                end->next=head;
}


void insert_start(){
        struct Node *temp;
        temp = (struct Node *)malloc(sizeof(struct Node));
        printf("Enter Data: ");
        scanf("%d",&temp->data);
        temp->next = head;
        head =temp;
        end->next=head;
}
```

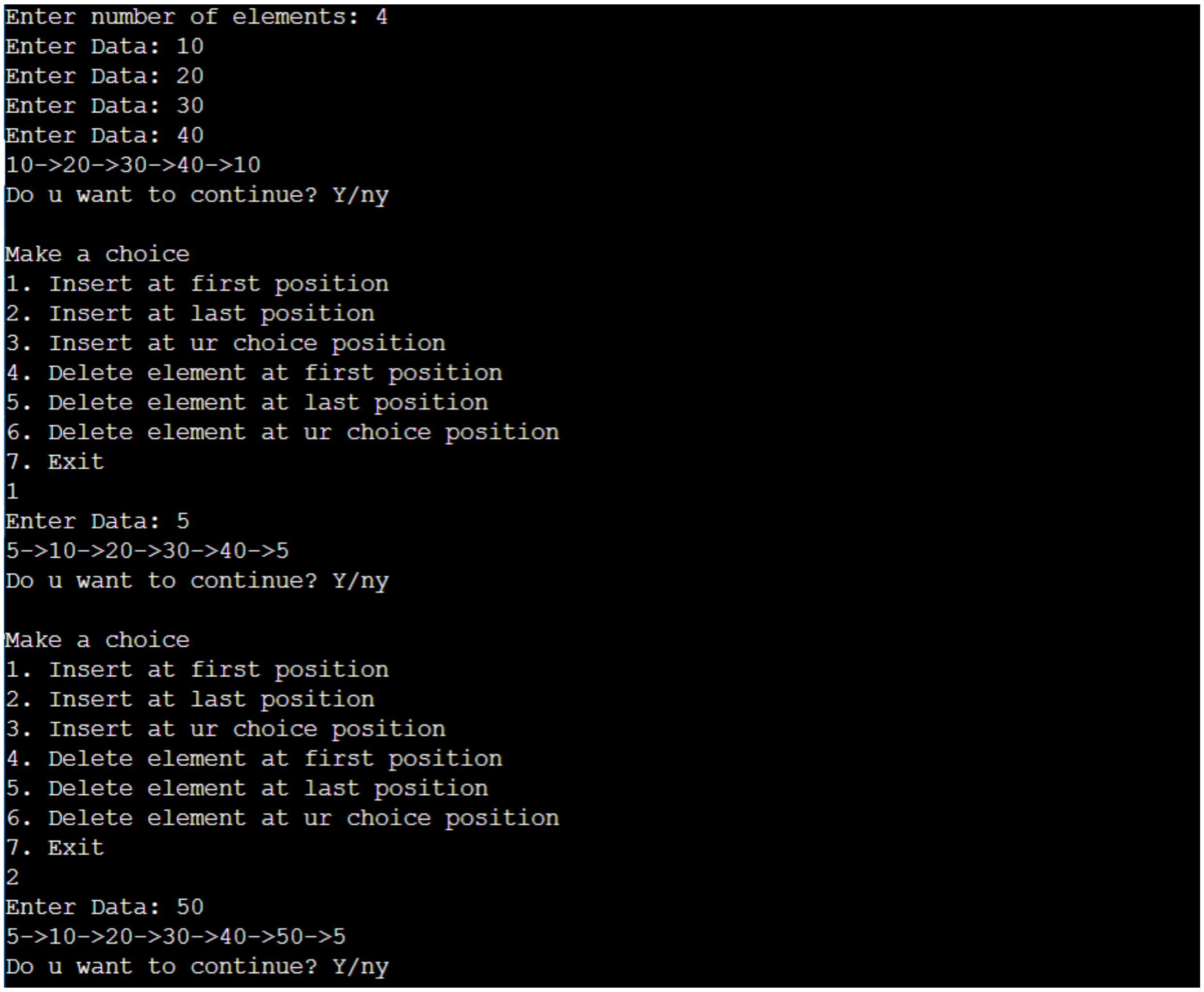
```
void insert_pos(){
    int pos,i;
    struct Node *t1, *t2, *temp;
    temp = (struct Node *)malloc(sizeof(struct Node));
    printf("Enter Data: ");
    scanf("%d",&temp->data);
    printf("Enter Position: ");
    scanf("%d",&pos);

    t2=head;
    for(i=0;i<pos-1;i++){
        t1=t2;
        t2=t1->next;
    }

    t1->next=temp;
    temp->next=t2;

}

void del_last(){
        struct Node *t1, *t2;
        t2=head;

        while(t2->next!=end){
                t1=t2;
                t2=t2->next;
        }

        t1->next = head;
        end=t1;
        free(t2);
}

void del_first(){
        struct Node *t1;
        t1=head;
        head = t1->next;
        end->next=head;
        free(t1);
}

void del_pos(){
        int i, pos;
        struct Node *t1, *t2;
        t2 =head;

        printf("Enter position: ");
        scanf("%d",&pos);

        for(i=0; i<pos-1;i++)
        {       t1=t2;
                t2 = t1->next;
        }
        t1->next=t2->next;
        free(t2);
}

void display(){
        struct Node *temp;
        temp=head;
        while (temp!=end){
                printf("%d->",temp->data);
                temp = temp->next;
        }
        printf("%d->%d",end->data,head->data);
    //printf("NULL");

}

int main(){
        int x,i,c;
        char con;
        //clrscr();
        printf("Enter number of elements: ");
        scanf("%d",&x);

        for (i=0;i<x; i++){
                create();
        }
        display();

        printf("\nDo u want to continue? Y/n");
        scanf("%s", &con);


        while ( con =='y'|| con == 'Y'){
        printf("\nMake a choice\n");
        printf("1. Insert at first position\n");
        printf("2. Insert at last position\n");
        printf("3. Insert at ur choice position\n");
        printf("4. Delete element at first position\n");
        printf("5. Delete element at last position\n");
        printf("6. Delete element at ur choice position\n");
        printf("7. Exit\n");
        scanf("%d",&c);
        switch(c){
                case 1: insert_start();
                        display();
                        break;

                case 2: insert_end();
                        display();
                        break;

                case 3: insert_pos();
                        display();
                        break;

                case 4: del_first();
                        display();
                        break;

                case 5: del_last();
                        display();
                        break;

                case 6: del_pos();
                        display();
                        break;

      case 7: exit(0);
                default: printf("Wrong choice:");
        }

        printf("\nDo u want to continue? Y/n");
        scanf("%s", &con);
        }
        //getch();
        return 0;

}
```

Output:

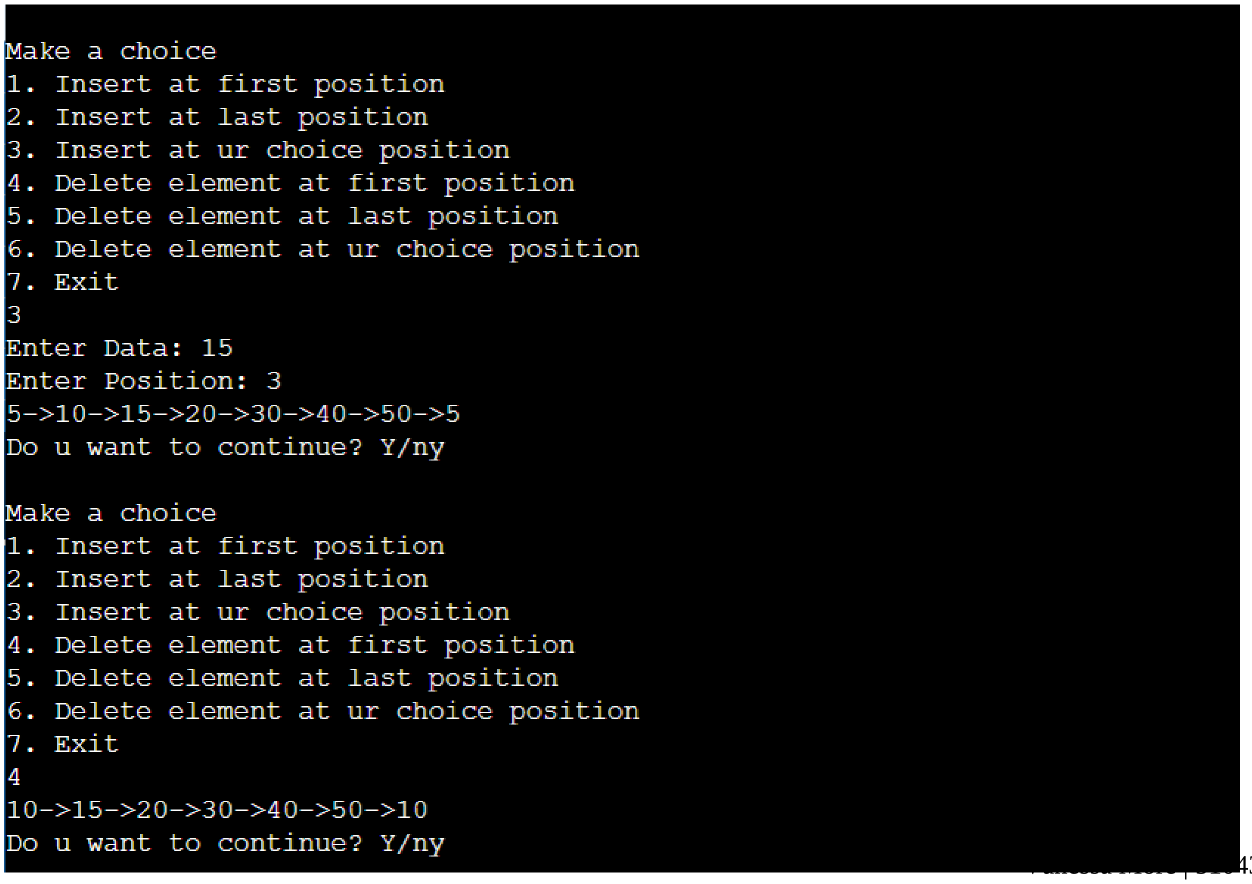
```
Enter number of elements: 4
Enter Data: 10
Enter Data: 20
Enter Data: 30'_ _
Enter Data: 40
10->20->30->40->10
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
1
Enter Data: 5
5->10->20->30->40->5
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
2
Enter Data: 50
5->10->20->30->40->50->5
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
3
Enter Data: 15
Enter Position: 3
5->10->15->20->30->40->50->5
Do u want to continue? Y/ny
```

```
Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
4
10->15->20->30->40->50->10
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
5
10->15->20->30->10
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
6
Enter position: 2
10->20->30->10
Do u want to continue? Y/nn
```

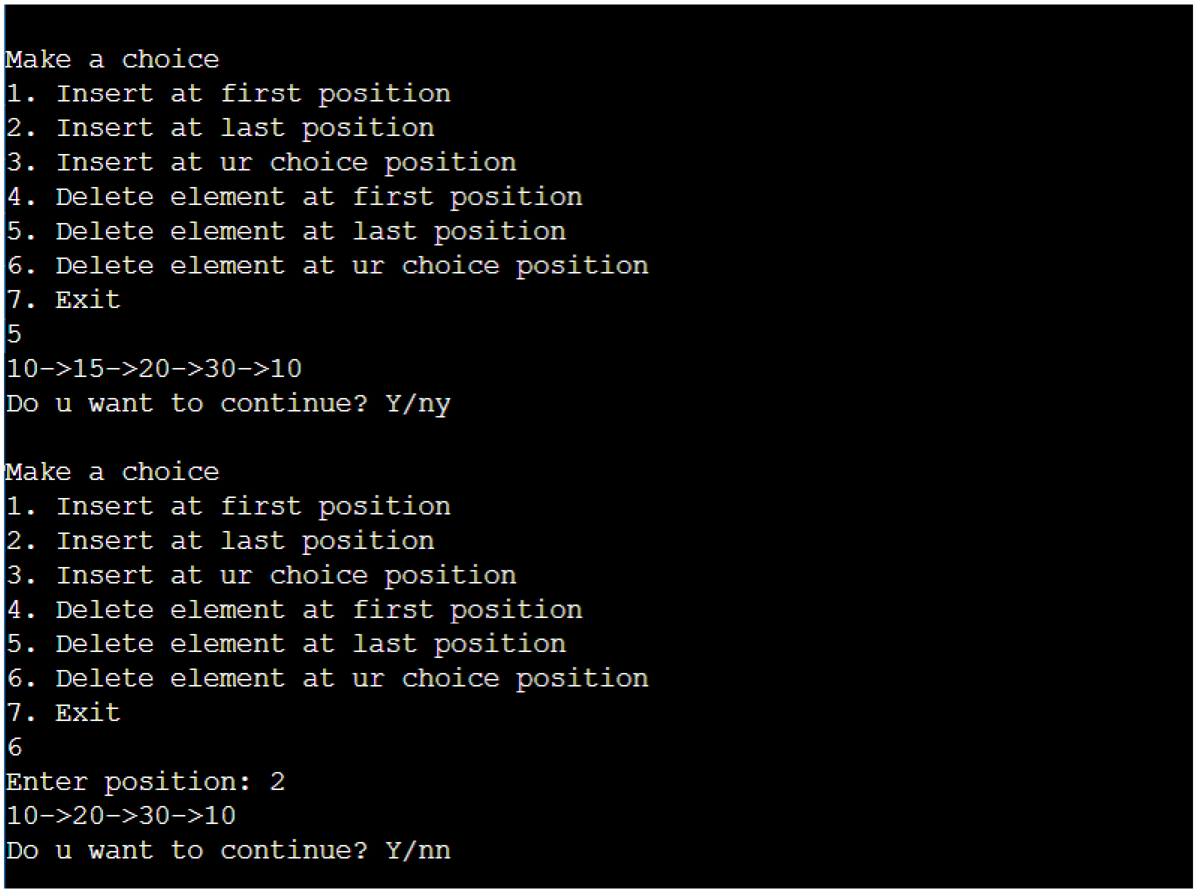
In Editor:



```
Enter number of elements: 4
Enter Data: 10
Enter Data: 20
Enter Data: 30
Enter Data: 40
10->20->30->40->10
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
1
Enter Data: 5
5->10->20->30->40->5
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
2
Enter Data: 50
5->10->20->30->40->50->5
Do u want to continue? Y/ny
```



```
Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
3
Enter Data: 15
Enter Position: 3
5->10->15->20->30->40->50->5
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
4
10->15->20->30->40->50->10
Do u want to continue? Y/ny
```



```
Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
5
10->15->20->30->10
Do u want to continue? Y/ny

Make a choice
1. Insert at first position
2. Insert at last position
3. Insert at ur choice position
4. Delete element at first position
5. Delete element at last position
6. Delete element at ur choice position
7. Exit
6
Enter position: 2
10->20->30->10
Do u want to continue? Y/nn
```

Q5. Represent polynomial as a singly linked list and write a menu driven program to perform addition and evaluation.

Program:

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

struct Node
{ int deg,coef;
 struct Node *next;

}*head=NULL, *end;

struct Node *newNode, *temp;

void create(){
        struct Node *temp;
        temp = (struct Node *)malloc(sizeof(struct Node));
        printf("Enter degree: ");
        scanf("%d",&temp->deg);
        printf("Enter coefficient: ");
        scanf("%d",&temp->coef);

        if(head==NULL){
                head=temp;
                end=temp;
        }
        else{
                end->next=temp;
                end=temp;
        }
}

void display(){
        struct Node *temp;
        temp=head;
        while (temp!=NULL){
                printf("%dx^%d + ",temp->coef,temp->deg);
                temp = temp->next;
        }
}

void evaluate(){
   struct Node *temp;
   int x, sum=0, power,i;
   printf("\nEnter value of variable x: ");
   scanf("%d",&x);
        temp=head;
```

```
while (temp!=NULL){
           power=1;
                for (i=1;i<=temp->deg;i++){
                  power = power * x;
                }

          sum = sum + temp->coef * power;
          temp=temp->next;
        }

        printf("\nEvaluation of polynomial = ");
        printf("%d",sum);
}

//Poly add

struct Node *create_add(struct Node *head)
{
int i,n;
   printf("\nEnter no. of terms: ");
   scanf("%d",&n);
   printf("\nEnter the terms in descending order of degree\n");
      for(i=0;i<n;i++)
      {
           newNode=(struct Node *)malloc(sizeof(struct Node));
           newNode->next=NULL;
           printf("coefficient: ");
           scanf("%d",&newNode->coef);
           printf("degree: ");
           scanf("%d",&newNode->deg);

        if(head==NULL)
        head=temp=newNode;

        else
        {
           temp->next=newNode;
           temp=newNode;
        }
      }
return head;
}

struct Node *add(struct Node *p1,struct Node *p2,struct Node *p3)
{
   struct Node *t1=p1,*t2=p2,*t3=p3;
   int i;

      while((t1!=NULL)&&(t2!=NULL))
      {
           newNode=(struct Node *)malloc(sizeof(struct Node));
           newNode->next=NULL;
```

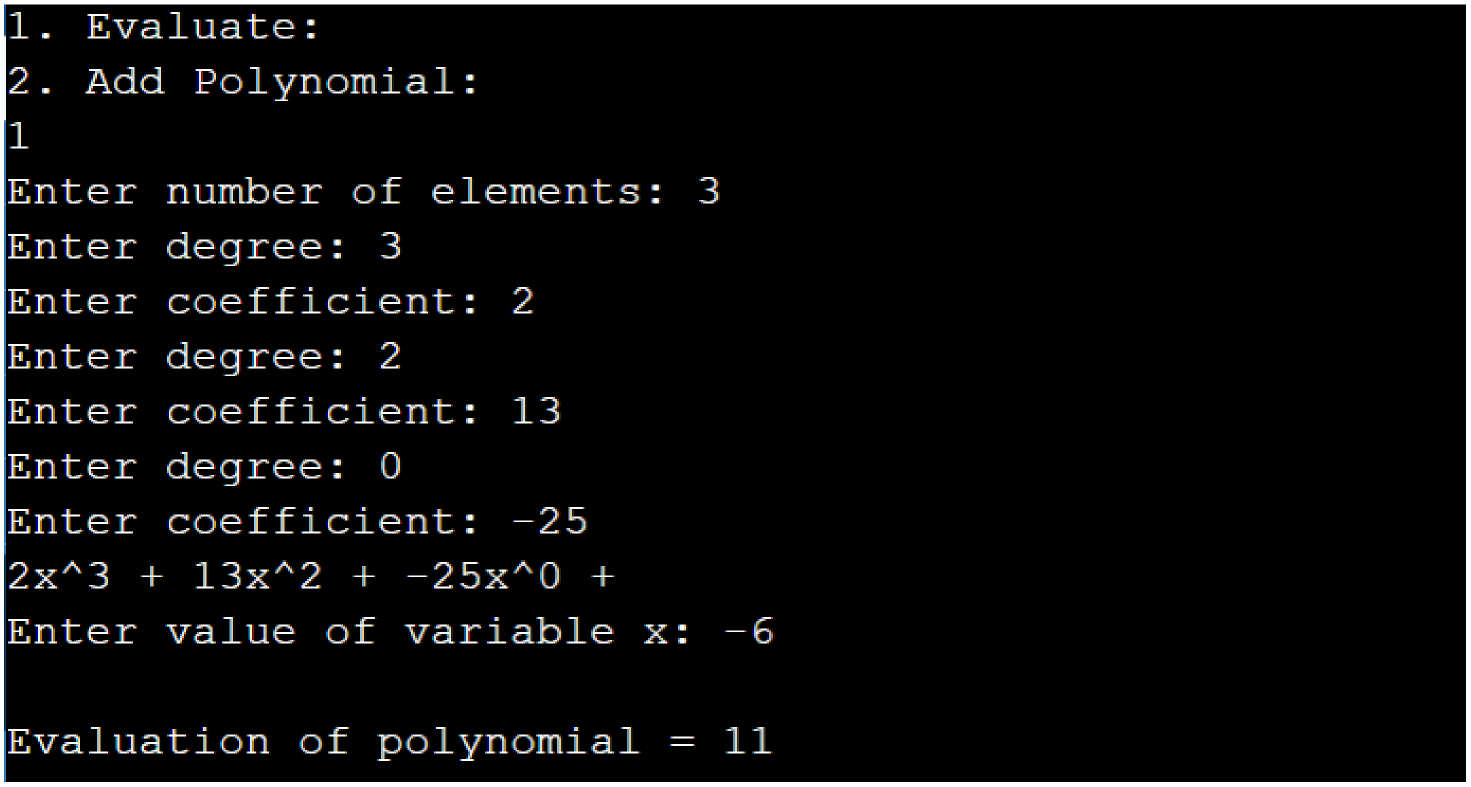
```
            if(t1->deg==t2->deg)
            {
                newNode->deg=t1->deg;
                newNode->coef=t1->coef+t2->coef;
                t1=t1->next;
                t2=t2->next;
            }
        else

            if(t1->deg>t2->deg)
            {
                newNode->deg=t1->deg;
                newNode->coef=t1->coef;
                t1=t1->next;
            }
            else
            {
                newNode->deg=t2->deg;
                newNode->coef=t2->coef;
                t2=t2->next;
            }

        if(p3==NULL)
            p3=t3=newNode;
        else
            {
                t3->next=newNode;
                t3=newNode;
            }
        }

        while(t1)
        {
            newNode=(struct Node *)malloc(sizeof(struct Node));
            newNode->next=NULL;
            newNode->deg=t1->deg;
            newNode->coef=t1->coef;
            t3->next=newNode;
            t3=newNode;
            t1=t1->next;
        }
        while(t2)
        {
            newNode=(struct Node *)malloc(sizeof(struct Node));
            newNode->next=NULL;
            newNode->deg=t2->deg;
            newNode->coef=t2->coef;
            t3->next=newNode;
            t3=newNode;
            t2=t2->next;
        }

    return p3;
}
```

```
void display_add(struct Node *head)
{
   struct Node *t=head;
      while(t)
      {
         printf("(%dx^%d)+",t->coef,t->deg);
         t=t->next;
      }
}


int main()
{
   int ch;
   int x,i;
   struct Node *p1=NULL,*p2=NULL,*p3=NULL;
   int n;
   char c;

   printf("1. Evaluate:\n2. Add Polynomial:\n");
   scanf("%d",&ch);


   switch (ch){
      case 1:
            printf("Enter number of elements: ");
                  scanf("%d",&x);
                  for (i=0;i<x; i++){
                           create();
                  }
                  display();

                  evaluate();
                  break;

      case 2:

            p1=create_add(p1);
            display_add(p1);
            p2=create_add(p2);
            display_add(p2);
            p3=add(p1,p2,p3);
            printf("\nthe addition is:");
            display_add(p3);
            break;

      default: printf("Wrong choice");

      }
   return 0;
}
```

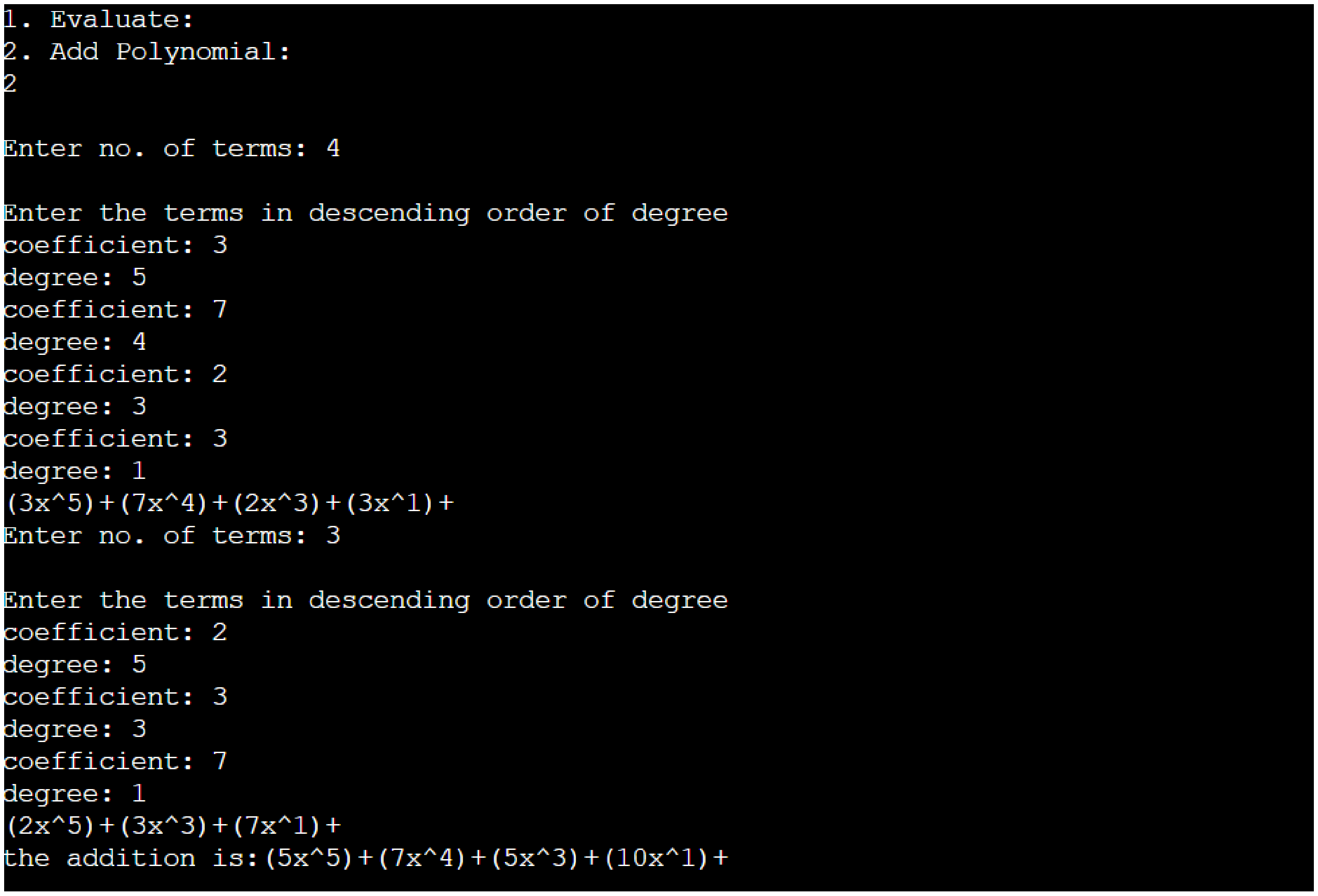
Output:

```
1. Evaluate:
2. Add Polynomial:
1
Enter number of elements: 3
Enter degree: 3
Enter coefficient: 2
Enter degree: 2
Enter coefficient: 13
Enter degree: 0
Enter coefficient: -25
2x^3 + 13x^2 + -25x^0 +
Enter value of variable x: -6

Evaluation of polynomial = 11


1. Evaluate:
2. Add Polynomial:
2

Enter no. of terms: 4

Enter the terms in descending order of degree
coefficient: 3
degree: 5
coefficient: 7
degree: 4
coefficient: 2
degree: 3
coefficient: 3
degree: 1
(3x^5)+(7x^4)+(2x^3)+(3x^1)+
Enter no. of terms: 3

Enter the terms in descending order of degree
coefficient: 2
degree: 5
coefficient: 3
degree: 3
coefficient: 7
degree: 1
(2x^5)+(3x^3)+(7x^1)+
the addition is:(5x^5)+(7x^4)+(5x^3)+(10x^1)+
```

In Editor:



```
1. Evaluate:
2. Add Polynomial:
1
Enter number of elements: 3
Enter degree: 3
Enter coefficient: 2
Enter degree: 2
Enter coefficient: 13
Enter degree: 0
Enter coefficient: -25
2x^3 + 13x^2 + -25x^0 +
Enter value of variable x: -6

Evaluation of polynomial = 11
```



```
1. Evaluate:
2. Add Polynomial:
2

Enter no. of terms: 4

Enter the terms in descending order of degree
coefficient: 3
degree: 5
coefficient: 7
degree: 4
coefficient: 2
degree: 3
coefficient: 3
degree: 1
(3x^5)+(7x^4)+(2x^3)+(3x^1)+
Enter no. of terms: 3

Enter the terms in descending order of degree
coefficient: 2
degree: 5
coefficient: 3
degree: 3
coefficient: 7
degree: 1
(2x^5)+(3x^3)+(7x^1)+
the addition is:(5x^5)+(7x^4)+(5x^3)+(10x^1)+
```

Progressive Education Society's

Modern College of Engineering, Pune

MCA Department

A.Y.2022-23

(310902) Data Structure and Algorithm Laboratory

****

Class: FY-MCA Shift /Div: A Batch: F2 Roll Number: 51043

Name: Vanessa Reetu Prashant More Assignment No: 3 Date of Implementation: 27/1/2023

****

Q1. Implement Stack using Array ADT.

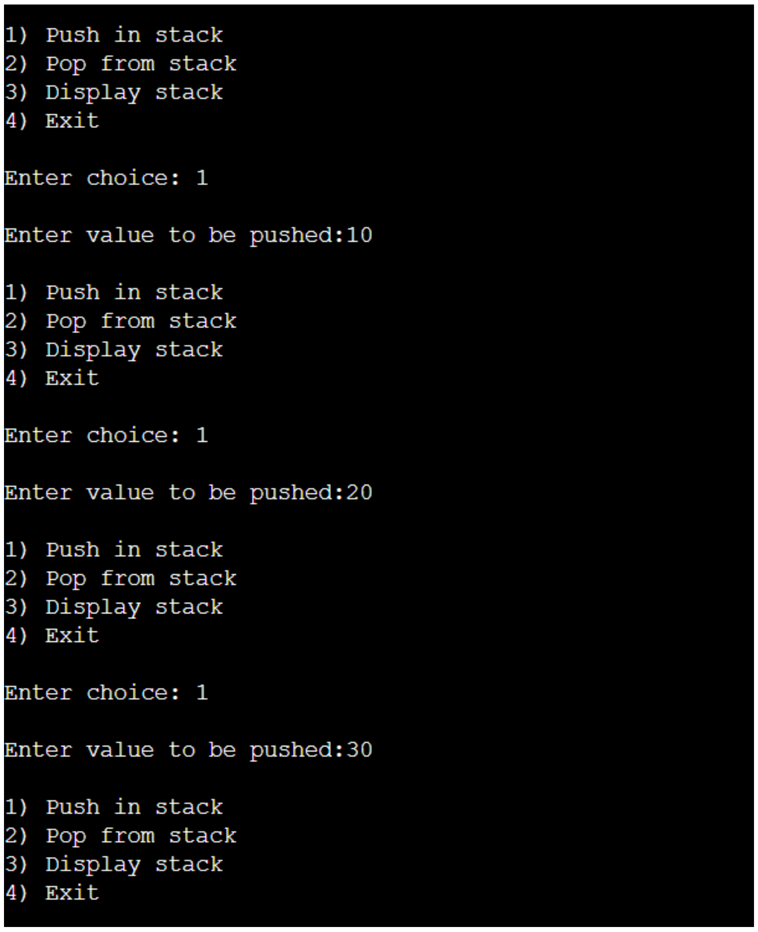
Program:

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

int stack[100], n=100, top=-1;
void push(int val) {
  if(top>=n-1)
  printf("Stack Overflow");
  else {
    top++;
    stack[top]=val;
  }
}
void pop() {
  if(top<=-1)
  printf("Stack Underflow");
  else {
     printf("The popped element is %d\n",stack[top]);
    top--;
  }
}
void display() {
  if(top>=0) {
    for(int i=top; i>=0; i--){
    printf("%d\n||\n\\/\n",stack[i]);
    }
    printf("NULL");
  } else
  printf("Stack is empty");
}
void main() {
  int ch, val;

  do {
   printf("\n1) Push in stack\n");
   printf("2) Pop from stack\n");
   printf("3) Display stack\n");
   printf("4) Exit\n");
    printf("\nEnter choice: ");
    scanf("%d",&ch);
    switch(ch) {
      case 1: {
        printf("\nEnter value to be pushed:");
        scanf("%d",&val);
        push(val);
        break;
      }
      case 2: {
        pop();
        display();
        break;
      }
      case 3: {
        display();
        break;
      }
      case 4: {
        printf("\nExit");
        break;
      }
      default: {
        printf("Invalid Choice");
      }
    }
  }while(ch!=4);
  getch();
}
```

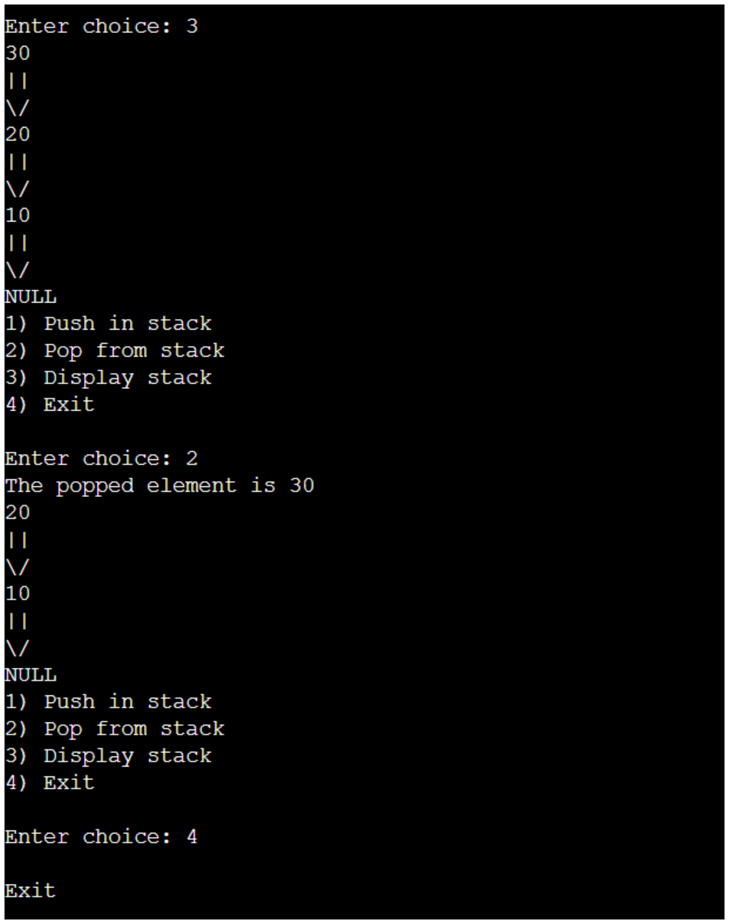
Output:

```
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 1

Enter value to be pushed:10

1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 1

Enter value to be pushed:20

1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 1
Enter value to be pushed:30

1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 3
30
||
\/
20
||
\/
10
||
\/
NULL
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 2
The popped element is 30
20
||
\/
10
||
\/
NULL
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 4

Exit
```

In Editor:



```
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 1

Enter value to be pushed:10

1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 1

Enter value to be pushed:20

1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 1

Enter value to be pushed:30

1) Push in stack
2) Pop from stack
3) Display stack
4) Exit
```



```
Enter choice: 3
30
||
\/
20
||
\/
10
||
\/
NULL
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 2
The popped element is 30
20
||
\/
10
||
\/
NULL
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 4

Exit
```

Q2. Implement Stack using Linked List.

Program:

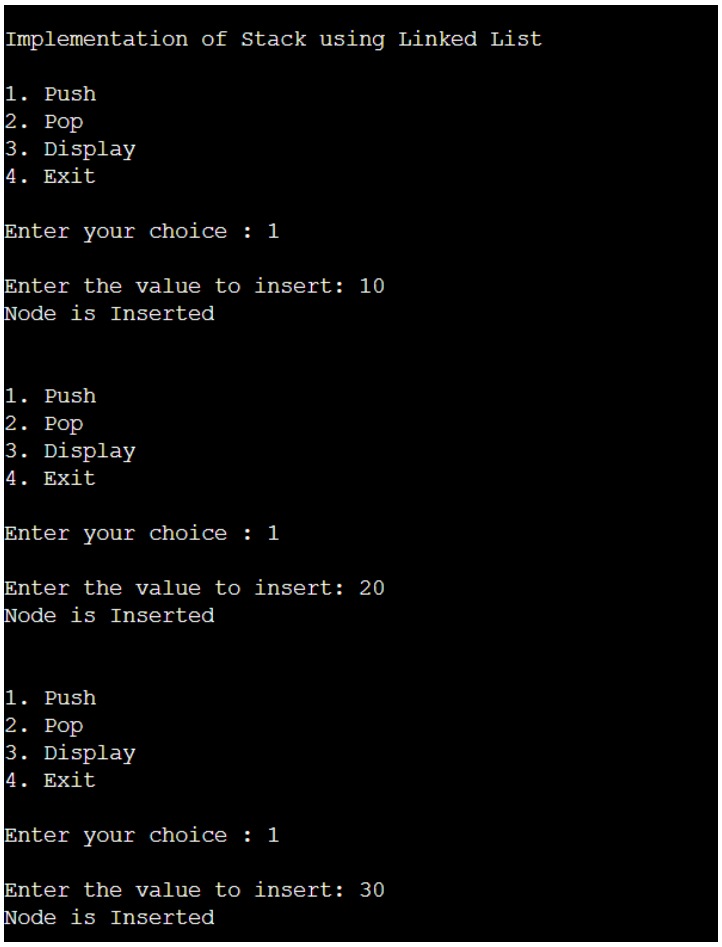
```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

struct Node {
    int data;
    struct Node *next;
}*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;
    printf("Node is Inserted\n\n");
}

int pop() {
    if (top == NULL) {
        printf("\nStack Underflow\n");
    } else {
        struct Node *temp = top;
        int temp_data = top->data;
        top = top->next;
        free(temp);
        return temp_data;
    }
}

void display() {
    if (top == NULL) {
        printf("\nStack Underflow\n");
    } else {
        printf("The stack is \n");
        struct Node *temp = top;
        while (temp->next != NULL) {
            printf("%d\n||\n\\/\n", temp->data);
            temp = temp->next;
        }
        printf("%d\n||\n\\/\nNULL", temp->data);
    }
}
```

```
int main() {
    int choice, value;
    printf("\nImplementation of Stack using Linked List\n");
    while (1) {
        printf("\n1. Push\n2. Pop\n3. Display\n4. Exit\n");
        printf("\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice) {
        case 1:
            printf("\nEnter the value to insert: ");
            scanf("%d", &value);
            push(value);
            break;
        case 2:
            printf("Popped element is :%d\n", pop());
            display();
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
            break;
        default:
            printf("\nWrong Choice\n");
        }
    }

    return 0;
}
```

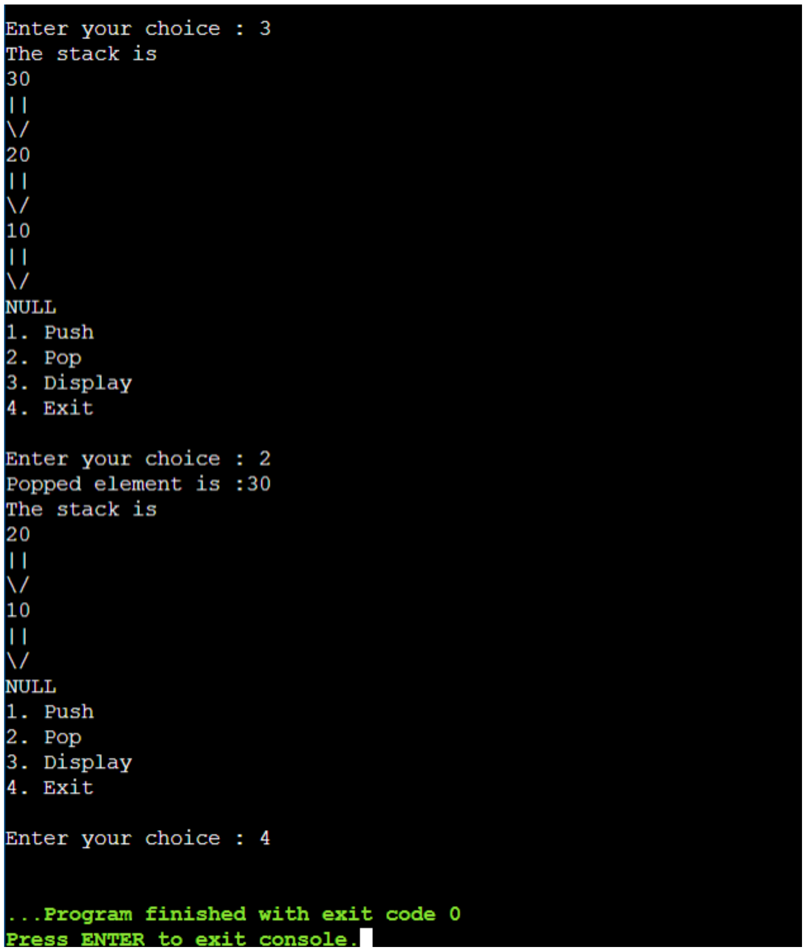
Output:

```
Implementation of Stack using Linked List

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

Enter your choice : 1

Enter the value to insert: 10
Node is Inserted


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

Enter your choice : 1
Enter the value to insert: 20
Node is Inserted


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

Enter your choice : 1

Enter the value to insert: 30
Node is Inserted


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

Enter your choice : 3
The stack is
30
||
\/
20
||
\/
10
||
\/
NULL
1. Push
2. Pop
3. Display
4. Exit

Enter your choice : 2
Popped element is :30
The stack is
20
||
\/
10
||
\/
NULL
1. Push
2. Pop
3. Display
4. Exit

Enter your choice : 4
```

In Editor:



```
Implementation of Stack using Linked List

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

Enter your choice : 1

Enter the value to insert: 10
Node is Inserted


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

Enter your choice : 1

Enter the value to insert: 20
Node is Inserted


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

Enter your choice : 1

Enter the value to insert: 30
Node is Inserted
```



```
Enter your choice : 3
The stack is
30
||
\/
20
||
\/
10
||
\/
NULL
1. Push
2. Pop
3. Display
4. Exit

Enter your choice : 2
Popped element is :30
The stack is
20
||
\/
10
||
\/
NULL
1. Push
2. Pop
3. Display
4. Exit

Enter your choice : 4


...Program finished with exit code 0
Press ENTER to exit console.
```

Q3. Implement stack as an ADT. Use this ADT to perform expression conversion. (Infix – Postfix)

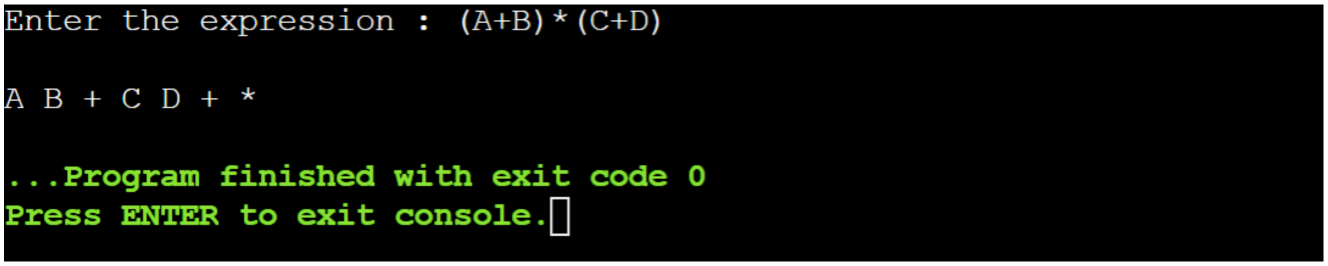
Program:

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

char stack[100];
int top = -1;

void
push (char x)
{
  stack[++top] = x;
}

char
pop ()
{
  if (top == -1)
    return -1;
  else
    return stack[top--];
}

int
priority (char x)
{
  if (x == '(')
    return 0;
  if (x == '+' || x == '-')
    return 1;
  if (x == '*' || x == '/')
    return 2;
  return 0;
}

int
main ()
{
  char exp[100];
  char *e, x;
  printf ("Enter the expression : ");
  scanf ("%s", exp);
  printf ("\n");
  e = exp;

  while (*e != '\0')
    {
      if (isalnum (*e))
    printf ("%c ", *e);
      else if (*e == '(')
    push (*e);
      else if (*e == ')')
    {
      while ((x = pop ()) != '(')
        printf ("%c ", x);
    }
      else
    {
      while (priority (stack[top]) >= priority (*e))
        printf ("%c ", pop ());
      push (*e);
    }
      e++;
    }

  while (top != -1)
    {
      printf ("%c ", pop ());
    }
  return 0;
}
```

Output:

```
Enter the expression : (A+B)*(C+D)
A B + C D + *
```

In Editor:



```
Enter the expression :  (A+B)*(C+D)

A B + C D + *

...Program finished with exit code 0
Press ENTER to exit console.
```

Q4. Write a program for Expression Evaluation using Stack.

Program:

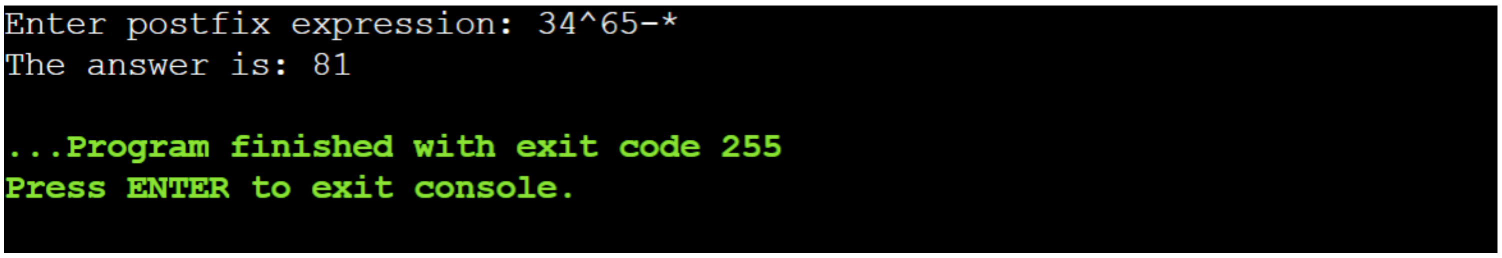
```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
#include<math.h>

#define max 10


int s[max];
int top =-1;

void push(int x){
    top ++;
    s[top]=x;
}

int pop(){
    int m;
    m=s[top];
    top--;
    return m;
}

void eval_postfix(char ch){

    int x,y,z;

    switch(ch){
        case '+':
            x = pop();
            y = pop();
            z = y + x;
            push(z);
            break;

        case '-':
            x = pop();
            y = pop();
            z = y - x;
            push(z);
            break;

        case '*':
            x = pop();
            y = pop();
            z = y * x;
            push(z);
            break;
```

```
 case '/':
         x = pop();
         y = pop();
         z = y / x;
         push(z);
         break;

     case '^':
         x = pop();
         y = pop();
         z = pow(y,x);
         push(z);
         break;
  }

}

void main(){

  char ch;
  int ans;
  printf("Enter postfix expression: ");
  while((ch=getchar())!='\n'){
     if(isdigit(ch)){
         push(ch-48);
     }else{
         eval_postfix(ch);
     }
  }

  ans = pop();
  printf("The answer is: %d",ans);
  getch();

}
```

Output:

```
Enter postfix expression: 34^65-*
The answer is: 81
```

In Editor:



```
Enter postfix expression: 34^65-*
The answer is: 81

...Program finished with exit code 255
Press ENTER to exit console.
```

Q5. Implement stack as an ADT. Use this ADT to perform expression conversion (Infix – Prefix).

Program:

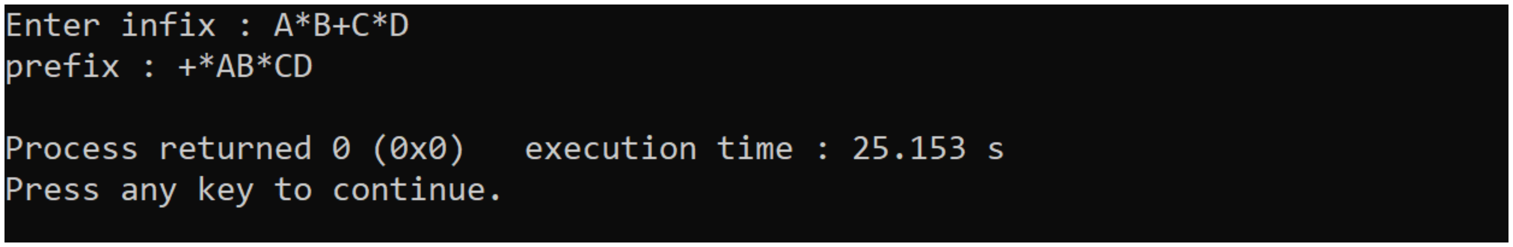
```
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<stdlib.h>

#define BLANK ' '
#define TAB '\t'
#define MAX 50

long int pop();
long int eval_pre();
char infix[MAX], prefix[MAX];
long int stack[MAX];
int top;
int isempty();
int white_space(char symbol);

void infix_to_prefix();
int priority(char symbol);
void push(long int symbol);
long int pop();

int main()
{
        long int value;
        top = -1;
        printf("Enter infix : ");
        gets(infix);
        infix_to_prefix();
        printf("prefix : %s\n",prefix);

        return 0;

}
void infix_to_prefix()
{
        int i,j,p,n;
        char next ;
        char symbol;
        char temp;
        n=strlen(infix);
        p=0;

        for(i=n-1; i>=0; i--)
        {
                symbol=infix[i];
                if(!white_space(symbol))
                {
                        switch(symbol)
                        {
                        case ')':
                                push(symbol);
                                break;
                        case '(':
                                while( (next=pop()) != ')')
                                        prefix[p++] = next;
                                break;
                        case '+':
                        case '-':
                        case '*':
                        case '/':
                        case '%':
                        case '^':
                                while( !isempty( ) &&  priority(stack[top])> priority(symbol) )
                                        prefix[p++] = pop();
                                push(symbol);
                                break;
                        default:
                             prefix[p++] = symbol;
                        }
                }
        }
        while(!isempty( ))
                prefix[p++] = pop();
        prefix[p] = '\0';

        for(i=0,j=p-1;i<j;i++,j--)
        {
                temp=prefix[i];
                prefix[i]=prefix[j];
                prefix[j]=temp;
        }
}

int priority(char symbol )
{
        switch(symbol)
        {
        case ')':
                return 0;
        case '+':
        case '-':
                return 1;
        case '*':
        case '/':
        case '%':
                return 2;
        case '^':
                return 3;
        default :
                 return 0;
        }
}
```

```
void push(long int symbol)
{
        if(top > MAX)
        {
                printf("Stack overflow\n");
                exit(1);
        }
        else
        {
                top=top+1;
                stack[top] = symbol;
        }
}

long int pop()
{
        if(top == -1 )
        {
                printf("Stack underflow \n");
                exit(2);
        }
        return (stack[top--]);
}

int isempty()
{
        if(top==-1)
                return 1;
        else
                return 0;
}

int white_space(char symbol)
{
        if(symbol==BLANK || symbol==TAB || symbol=='\0')
                return 1;
        else
                return 0;
}
```

Output:

```
Enter infix : A*B+C*D
prefix : +*AB*CD
```

In Editor:



```
Enter infix : A*B+C*D
prefix : +*AB*CD

Process returned 0 (0x0)   execution time : 25.153 s
Press any key to continue.
```

Q6. Implement circular queue using arrays.

Program:

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

int cqueue_arr[MAX];
int front = -1;
int rear = -1;


void enqueue(int item)
{
        if((front == 0 && rear == MAX-1) || (front == rear+1))
        {
           printf("Queue Overflow \n");
           return;
        }
        if (front == -1)
        {
           front = 0;
           rear = 0;
        }
        else
        {
           if(rear == MAX-1)
                rear = 0;
           else
                rear = rear+1;
        }
        cqueue_arr[rear] = item ;
}

void dequeue()
{
        if (front == -1)
        {
           printf("Queue Underflow\n");
           return ;
        }
        printf("Element deleted from queue is : %d\n",cqueue_arr[front]);
        if(front == rear)
        {
           front = -1;
           rear=-1;
        }
        else
        {
           if(front == MAX-1)
                front = 0;
           else
                front = front+1;
        }
}
```

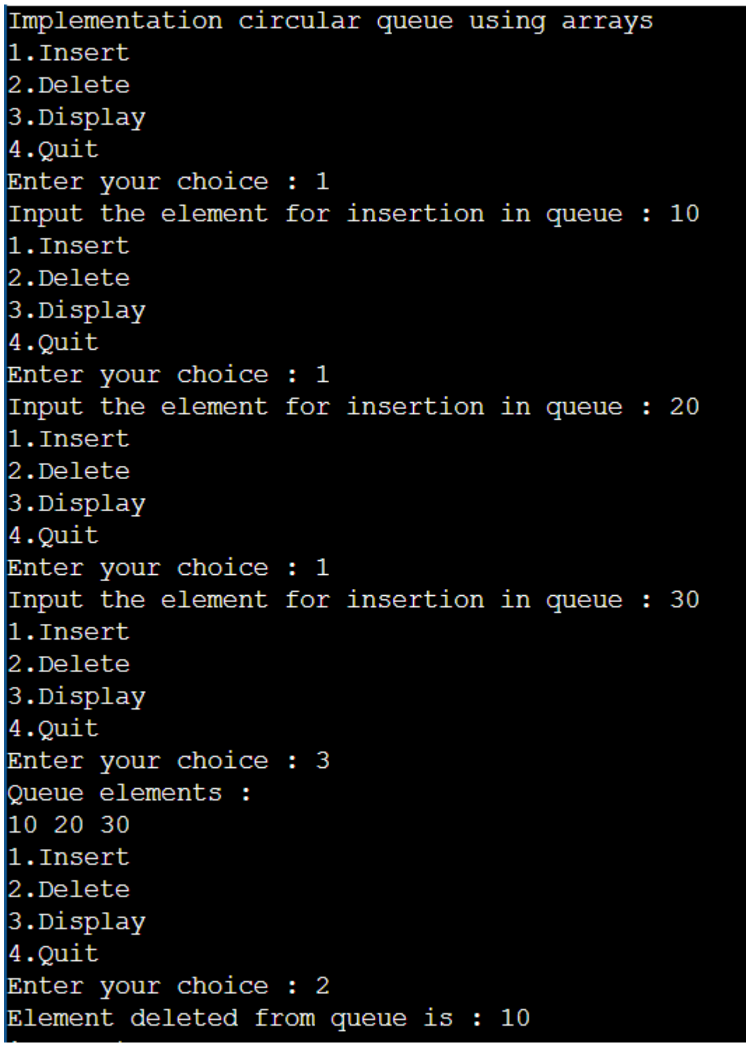
```
void display()
{
        int front_pos = front,rear_pos = rear;
        if(front == -1)
        {
          printf("Queue is empty\n");
          return;
        }
        printf("Queue elements :\n");
        if( front_pos <= rear_pos )
          while(front_pos <= rear_pos)
          {
                printf("%d ",cqueue_arr[front_pos]);
                front_pos++;
          }
        else
        {
          while(front_pos <= MAX-1)
          {
                printf("%d ",cqueue_arr[front_pos]);
                front_pos++;
          }
          front_pos = 0;
          while(front_pos <= rear_pos)
          {
                printf("%d ",cqueue_arr[front_pos]);
                front_pos++;
          }
        }
        printf("\n");
}

int main()
{
        int choice,item;

        printf("Implementation circular queue using arrays\n");
        do
        {
          printf("1.Insert\n");
          printf("2.Delete\n");
          printf("3.Display\n");
          printf("4.Quit\n");

          printf("Enter your choice : ");
          scanf("%d",&choice);

          switch(choice)
          {
                case 1 :
                        printf("Input the element for insertion in queue : ");
                        scanf("%d", &item);
                        enqueue(item);
                        break;
                case 2 :
                        dequeue();
                        break;
                    case 3:
                        display();
                        break;
                    case 4:
                        break;
                        default:
                        printf("Wrong choice\n");
                }
            }while(choice!=4);

            return 0;
        }
```

Output:

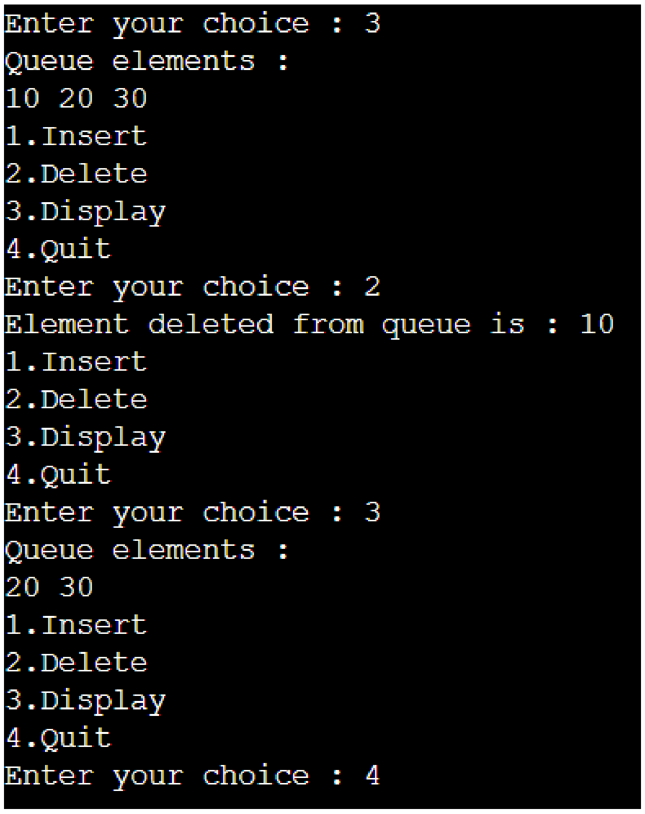
```
Implementation circular queue using arrays
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 10
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 20
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 30
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 3
Queue elements :
10 20 30
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 2
Element deleted from queue is : 10
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 3
```

```
Queue elements :
20 30
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 4
```

In Editor:



```
Implementation circular queue using arrays
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 10
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 20
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 30
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 3
Queue elements :
10 20 30
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 2
Element deleted from queue is : 10
```



```
Enter your choice : 3
Queue elements :
10 20 30
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 2
Element deleted from queue is : 10
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 3
Queue elements :
20 30
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 4
```

Q7. Implement job scheduling algorithm using queue.

Program:

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

int pqPriority[maxsize];
int pqAt[maxsize];
int pqBt[maxsize];
int pqPid[maxsize];
int pqWt[maxsize];
int pqSt[maxsize];
int pqTt[maxsize];
int front = -1, rear = -1;

void enqueue()
{
   int pid;
   int bt;
   int priority;
   int at;

      printf("\nEnter pid: ");
      scanf("%d",&pid);

      printf("Enter Burst time: ");
      scanf("%d",&bt);

      printf("Enter arrival Time: ");
      scanf("%d",&at);

      printf("Enter priority: ");
      scanf("%d",&priority);



   if(rear == maxsize-1)
   {
      printf("\nOVERFLOW\n");
      return;
   }
   if(front == -1 && rear == -1)
   {
      front = 0;
      rear = 0;
   }
   else
   {
      rear = rear+1;
   }
```

```
    pqPid[rear] = pid;
    pqBt[rear]=bt;
    pqPriority[rear]=priority;
    pqAt[rear]=at;
    printf("\nValue inserted ");

}
void dequeue()
{
    int item;
    if (front == -1 || front > rear)
    {
        printf("\nUNDERFLOW\n");
        return;

    }
    else
    {
        item = pqPid[front];
        if(front == rear)
        {
            front = -1;
            rear = -1 ;
        }
        else
        {
            front = front + 1;
        }
        printf("\nProcess P%d has been dequed ",item);
    }

}

void display()
{
    int i;
    if(rear == -1)
    {
        printf("\nEmpty queue\n");
    }
    else
    {   printf("\nprinting values .....\n");

        printf("\nPID\tBurstTime\tArrivalTime\tPriority");
        for(i=front;i<=rear;i++)
        {
            printf("\nP%d\t\t%d\t\t%d\t\t%d\n",pqPid[i],pqBt[i],pqAt[i],pqPriority[i]);
        }
    }
}
```

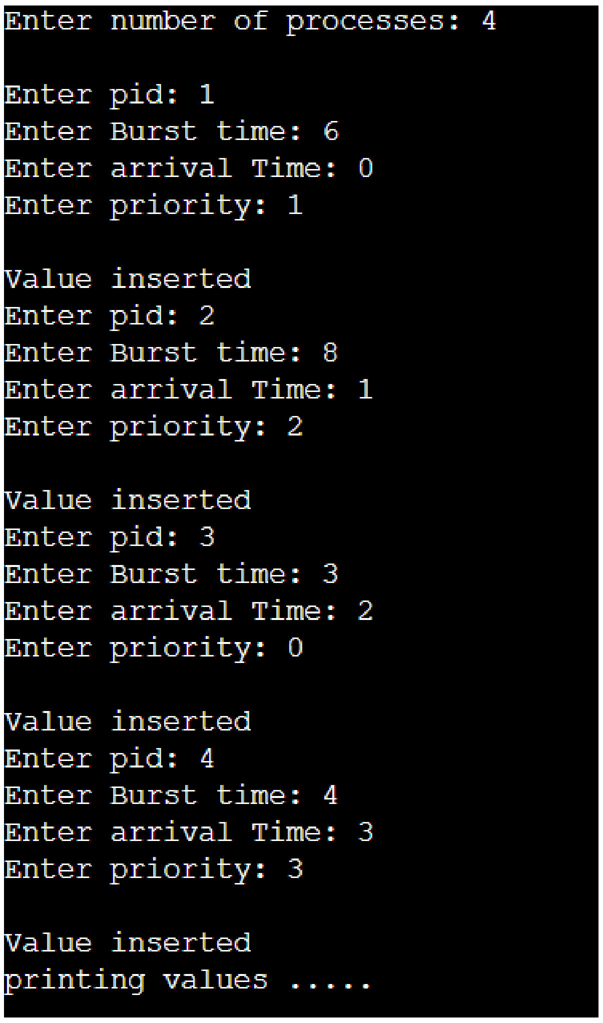
```
void startTime(){
   int i=0;
   pqSt[0]=0;
   for(i=1;i<rear+1;i++){
         pqSt[i]=pqBt[i-1] + pqSt[i-1];
   }

   pqSt[i]=pqBt[i-1]+pqSt[i-1];

   //display start time
   /*for(i=0;i<rear+1;i++){
         printf("%d\t",pqSt[i]);

   }*/

}

void turnAroundTime(){
   int i=0;
   float avgtt,sum=0;
   pqTt[0]=pqBt[0];
   for(i=1;i<rear+1;i++){
         pqTt[i]=pqSt[i+1] - pqAt[i];
   }

   /*printf("\n");
   for(i=0;i<rear+1;i++){
         printf("%d\t",pqTt[i]);
   }*/

   for(i=0;i<rear+1;i++){
         sum = sum + pqTt[i];
   }

   avgtt = sum / i;

   printf("\nAverage turnAroundTime= %f",avgtt);
}


void waitingTime(){
   int i=0;
   float avgwt,sum=0;
   for(i=0;i<rear+1;i++){
         pqWt[i]=pqSt[i] - pqAt[i];
   }
```

```
    //display start time
    /*for(i=0;i<rear+1;i++){
            printf("%d\t",pqWt[i]);

    }*/

    for(i=0;i<rear+1;i++){
            sum = sum + pqWt[i];
    }

    avgwt = sum / i;

    printf("\nAverage waitingTime= %f",avgwt);

}
int main(){

    int n, i = 0;

    printf("Enter number of processes: ");
    scanf("%d",&n);

    for (i=0;i<n;i++){
        enqueue();
    }

    display();
    startTime();
    waitingTime();
    turnAroundTime();

    printf("\nDequeing Process.... ");
    dequeue();

    display();

    return 0;
}
```

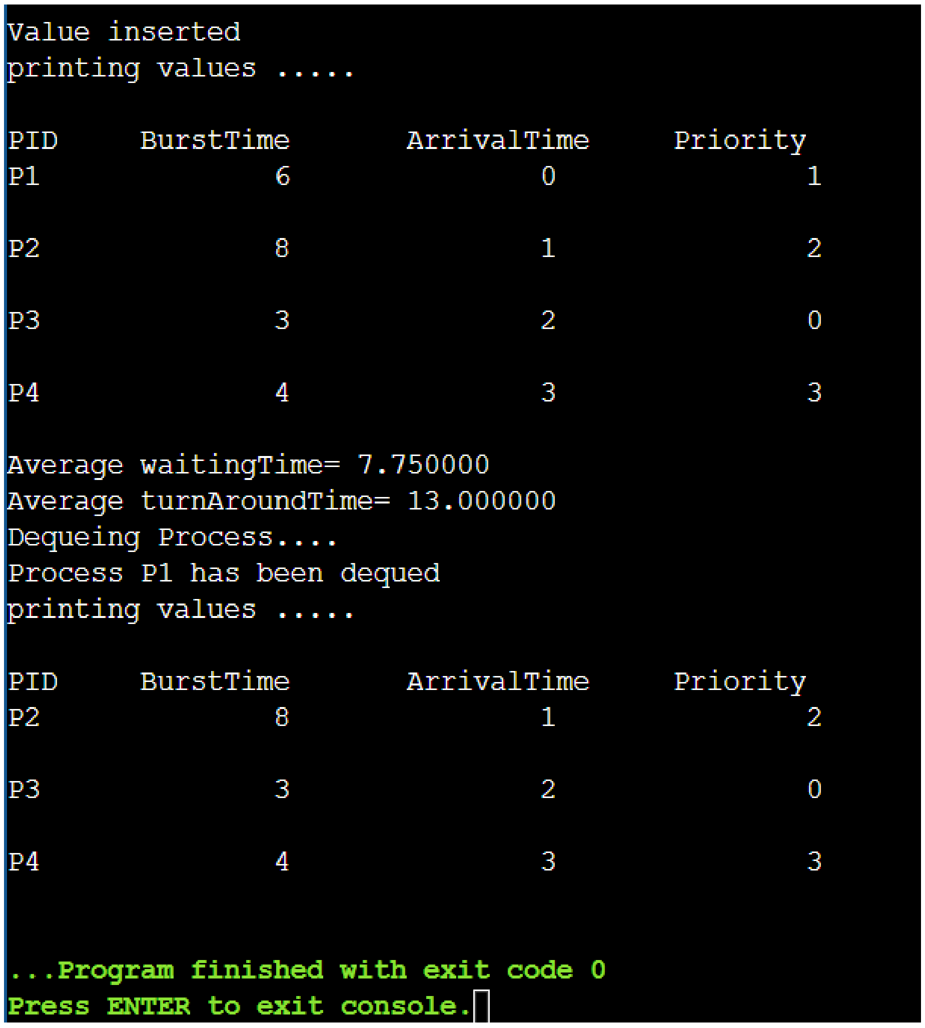
Output:

```
Enter number of processes: 4

Enter pid: 1
Enter Burst time: 6
Enter arrival Time: 0
Enter priority: 1

Value inserted
Enter pid: 2
Enter Burst time: 8
Enter arrival Time: 1
Enter priority: 2

Value inserted
Enter pid: 3
Enter Burst time: 3
Enter arrival Time: 2
Enter priority: 0

Value inserted
Enter pid: 4
Enter Burst time: 4
Enter arrival Time: 3
Enter priority: 3

Value inserted
printing values .....

PID    BurstTime      ArrivalTime     Priority
P1          6             0             1

P2          8             1             2

P3          3             2             0

P4          4             3             3

Average waitingTime= 7.750000
Average turnAroundTime= 13.000000
Dequeing Process....
Process P1 has been dequed
printing values .....

PID    BurstTime      ArrivalTime     Priority
P2          8             1             2

P3          3             2             0

P4          4             3             3
```

In Editor:



```
Enter number of processes: 4

Enter pid: 1
Enter Burst time: 6
Enter arrival Time: 0
Enter priority: 1

Value inserted
Enter pid: 2
Enter Burst time: 8
Enter arrival Time: 1
Enter priority: 2

Value inserted
Enter pid: 3
Enter Burst time: 3
Enter arrival Time: 2
Enter priority: 0

Value inserted
Enter pid: 4
Enter Burst time: 4
Enter arrival Time: 3
Enter priority: 3

Value inserted
printing values .....
```



```
Value inserted
printing values .....

PID     BurstTime       ArrivalTime     Priority
P1              6               0               1

P2              8               1               2

P3              3               2               0

P4              4               3               3

Average waitingTime= 7.750000
Average turnAroundTime= 13.000000
Dequeing Process....
Process P1 has been dequed
printing values .....

PID     BurstTime       ArrivalTime     Priority
P2              8               1               2

P3              3               2               0

P4              4               3               3


...Program finished with exit code 0
Press ENTER to exit console.
```

Progressive Education Society's

Modern College of Engineering, Pune

MCA Department

A.Y.2022-23

(310902) Data Structure and Algorithm Laboratory

****

Class: FY-MCA Shift /Div: A Batch: F2 Roll Number: 51043

Name: Vanessa Reetu Prashant More Assignment No: 4 Date of Implementation: 9/2/2023

****

Q1. Create binary tree and perform recursive traversals.

Program:

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

struct node{
        int data;
        struct node *leftChild;
        struct node *rightChild;
};

struct node *root=NULL;

void insert(int data){
        struct node *tempNode= (struct node*)malloc (sizeof(struct node));
        struct node *current;
        struct node *parent;

        tempNode->data = data;
        tempNode->leftChild =NULL;
        tempNode->rightChild=NULL;

        //if tree is empty
        if(root==NULL){
                root = tempNode;
        }else{
                current=root;
                parent=NULL;

                while(1){
                        parent =current;

                        //go to left of the tree
                        if(data < parent->data){
                                current = current->leftChild;

                                //insert to the left
```

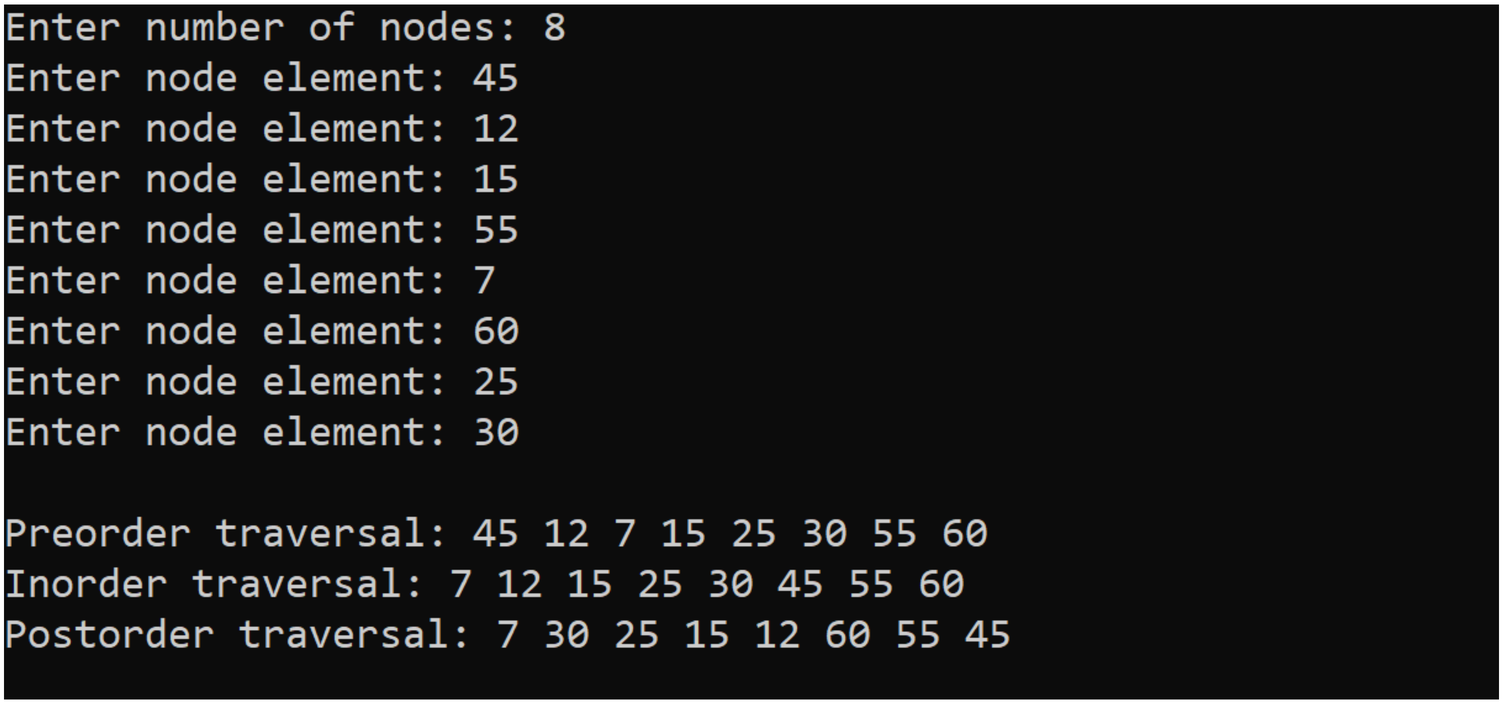
```
				if(current ==NULL){
						parent->leftChild=tempNode;
						return;
					}
				}
				//else go to right of the tree
				else{
					current = current->rightChild;

					//insert to the right
					if (current == NULL){
						parent->rightChild = tempNode;
						return;
					}
				}
			}
		}
	}

	//pre order traversal method

	void pre_order_traversal(struct node* root){

		if (root!=NULL){
			printf("%d ",root->data);
			pre_order_traversal(root->leftChild);
			pre_order_traversal(root->rightChild);
		}
	}

	//in order traversal method

	void in_order_traversal(struct node* root){

		if (root!=NULL){

			in_order_traversal(root->leftChild);
			printf("%d ",root->data);
			in_order_traversal(root->rightChild);
		}
	}

	//post order traversal method

	void post_order_traversal(struct node* root){

		if (root!=NULL){

			post_order_traversal(root->leftChild);
			post_order_traversal(root->rightChild);
			printf("%d ",root->data);
		}
	}
```

```
int main(){

        int i;
        int n;
        int array[50];
        struct node * temp;

        printf("Enter number of nodes: ");
        scanf("%d",&n);

        for (i=0;i<n;i++){
           printf("Enter node element: ");
           scanf("%d",&array[i]);
        }



        //create tree by inserting data one by one

        for(i=0; i<n ; i++)
                insert(array[i]);



        printf("\nPreorder traversal: ");
        pre_order_traversal(root);

        printf("\nInorder traversal: ");
        in_order_traversal(root);

        printf("\nPostorder traversal: ");
        post_order_traversal(root);

        getch();
        return 0;
}
```

Output:

```
Enter number of nodes: 8
Enter node element: 45
Enter node element: 12
Enter node element: 15
Enter node element: 55
Enter node element: 7
Enter node element: 60
Enter node element: 25
Enter node element: 30

Preorder traversal: 45 12 7 15 25 30 55 60
Inorder traversal: 7 12 15 25 30 45 55 60
Postorder traversal: 7 30 25 15 12 60 55 45
```

In Editor:



```
Enter number of nodes: 8
Enter node element: 45
Enter node element: 12
Enter node element: 15
Enter node element: 55
Enter node element: 7
Enter node element: 60
Enter node element: 25
Enter node element: 30

Preorder traversal: 45 12 7 15 25 30 55 60
Inorder traversal: 7 12 15 25 30 45 55 60
Postorder traversal: 7 30 25 15 12 60 55 45
```

Q2. . Create binary tree. Find height of the tree and print leaf nodes. Find mirror image, print original and mirror image using level-wise printing.

Program:

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

struct node{
        int data;
        struct node *leftChild;
        struct node *rightChild;
};

struct node *root=NULL;

void insert(int data){
        struct node *tempNode= (struct node*)malloc (sizeof(struct node));
        struct node *current;
        struct node *parent;

        tempNode->data = data;
        tempNode->leftChild =NULL;
        tempNode->rightChild=NULL;

        //if tree is empty

        if(root==NULL){
                root = tempNode;
        }else{
                current=root;
                parent=NULL;

                while(1){
                        parent =current;

                        //go to left of the tree
                        if(data < parent->data){
                                current = current->leftChild;

                                //insert to the left

                                if(current ==NULL){
                                        parent->leftChild=tempNode;
                                        return;
                                }
                        }
                        //else go to right of the tree
                        else{
                                current = current->rightChild;

                                //insert to the right
                                if (current == NULL){
```

```
                                                parent->rightChild = tempNode;
                                                return;
                                        }
                                }
                        }
                }
        }

        //in order traversal method

        void in_order_traversal(struct node* root){

                if (root!=NULL){

                        in_order_traversal(root->leftChild);
                        printf("%d ",root->data);
                        in_order_traversal(root->rightChild);
                }
        }


        //height of tree
        int height(struct node* root){
           int lheight, rheight;
           if (root==NULL)
              return -1;
           else{

              lheight = height(root->leftChild);
              rheight = height(root->rightChild);
                 if(lheight>rheight)
                    return lheight+1;
                 else
                    return rheight+1;
           }
        }


        //mirror
        void mirror(struct node* root){
           struct node *temp;
           if(root!=NULL){

           temp=root->leftChild;
           root->leftChild = root->rightChild;
           root->rightChild = temp;
           mirror(root->leftChild);
           printf("%d ",root->data);
           mirror(root->rightChild);
           }
           //return root;

        }
```

```
//Mirror tree Level wise tree
void mirrorLevel(struct node* root, int currentLevel, int level) {

  if(root == NULL) {
      return;
  }
  if(currentLevel == level) {
      printf(" %d ", root->data);
      return;
  }

  mirrorLevel(root->leftChild, currentLevel+1, level);
  mirrorLevel(root->rightChild, currentLevel+1, level);
}

void mirrorlevelwise(struct node* root){
    /* Find the height of tree */
    int i, h = height(root);
    /* Iterate from level 0 to height-1 and
 print one level at a time */
    for(i = 0; i <= h; i++){
     mirrorLevel(root, 0, i);
     printf("\n");
    }
}


//Original Tree level Wise
void printNodesAtLevel(struct node* root, int currentLevel, int level) {

  if(root == NULL) {
      return;
  }
  if(currentLevel == level) {
      printf(" %d ", root->data);
      return;
  }

  printNodesAtLevel(root->rightChild, currentLevel+1, level);
  printNodesAtLevel(root->leftChild, currentLevel+1, level);
}

/* Printed nodes of binary tree level wise */
void levelwise(struct node* root){
    /* Find the height of tree */
    int i, h = height(root);
    /* Iterate from level 0 to height-1 and
 print one level at a time */
    for(i = 0; i <= h; i++){
     printNodesAtLevel(root, 0, i);
     printf("\n");
    }
}
```

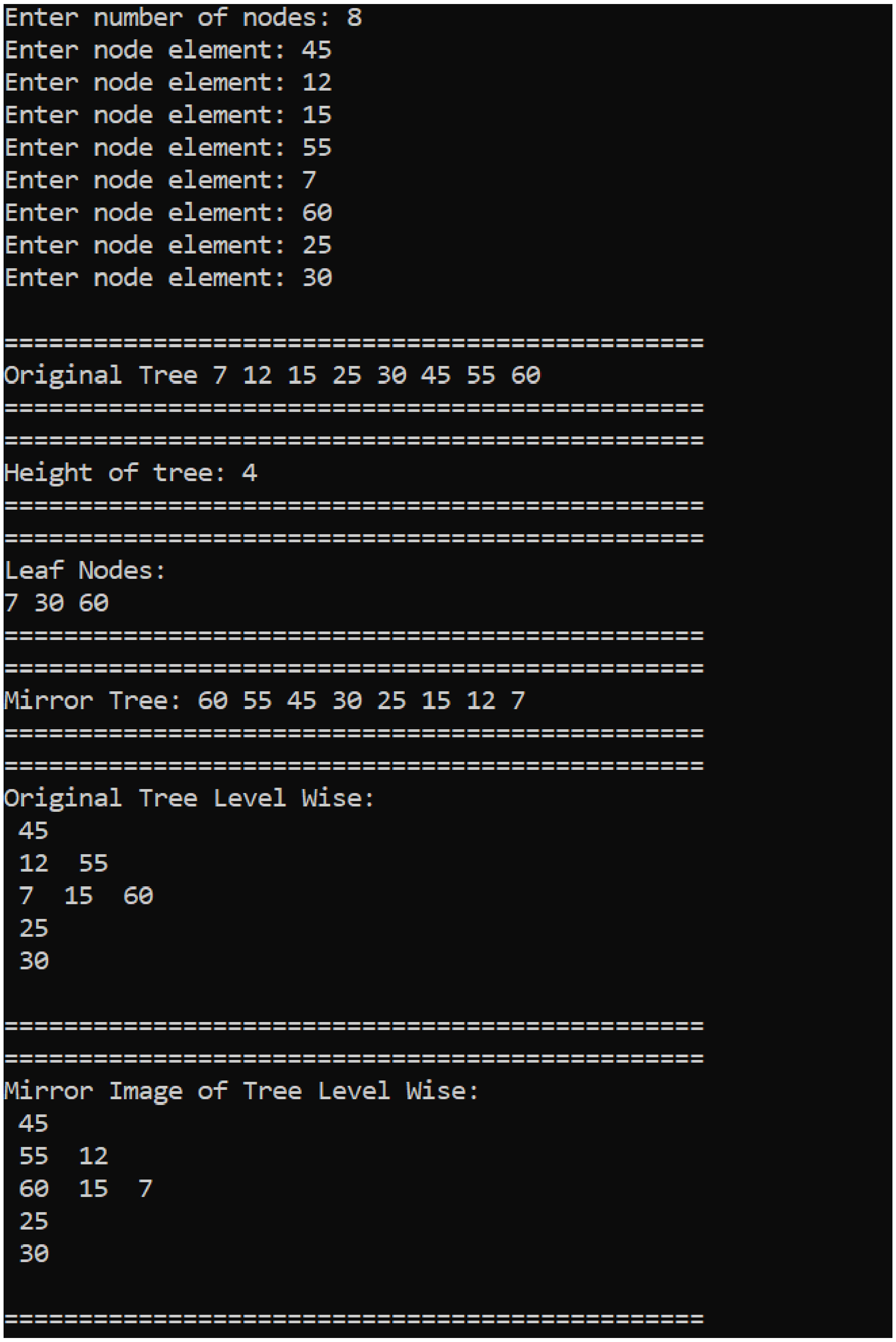
```
//Print leaf nodes
void printLeafNodes(struct node* root){
   if (root==NULL)
      return ;

   if (root->leftChild==NULL && root->rightChild==NULL){
      printf("%d ",root->data);
      return;
   }

   if(root->leftChild)
      printLeafNodes(root->leftChild);

   if(root->rightChild)
      printLeafNodes(root->rightChild);

}

int main(){

         int i;
         int n;
         //int array[8] = {45,12,15,55,7,60,25,30};
         int array[50];
         struct node * temp;
         //clrscr();

         printf("Enter number of nodes: ");
         scanf("%d",&n);

         for (i=0;i<n;i++){
            printf("Enter node element: ");
            scanf("%d",&array[i]);
         }

         //create tree by inserting data one by one
         for(i=0; i<n ; i++)
                  insert(array[i]);

         printf("\n==================================================");
         printf("\nOriginal Tree ");
         in_order_traversal(root);
         printf("\n==================================================");


         printf("\n==================================================");
         printf("\nHeight of tree: %d",height(root));
         printf("\n==================================================");

         printf("\n==================================================");
         printf("\nLeaf Nodes: \n");
         printLeafNodes(root);
         printf("\n==================================================");
```

```
        printf("\n================================================");
        printf("\nMirror Tree: ");
        mirror(root);
        printf("\n================================================");

        printf("\n================================================");
        printf("\nOriginal Tree Level Wise:\n");
        levelwise(root);
        printf("\n================================================");

        printf("\n================================================");
        printf("\nMirror Image of Tree Level Wise:\n");
        mirrorlevelwise(root);
        printf("\n================================================");


        getch();
        return 0;
}
```

Output:

```
Enter number of nodes: 8
Enter node element: 45
Enter node element: 12
Enter node element: 15
Enter node element: 55
Enter node element: 7
Enter node element: 60
Enter node element: 25
Enter node element: 30

================================================
Original Tree 7 12 15 25 30 45 55 60
================================================
================================================
Height of tree: 4
================================================
================================================
Leaf Nodes:
7 30 60
================================================
================================================
Mirror Tree: 60 55 45 30 25 15 12 7
================================================
================================================
Original Tree Level Wise:
 45
 12  55
 7  15  60
 25
 30

================================================
==================================================
Mirror Image of Tree Level Wise:
 45
 55  12
 60  15  7
 25
 30

==================================================
```

In Editor:



```
Enter number of nodes: 8
Enter node element: 45
Enter node element: 12
Enter node element: 15
Enter node element: 55
Enter node element: 7
Enter node element: 60
Enter node element: 25
Enter node element: 30

==============================================
Original Tree 7 12 15 25 30 45 55 60
==============================================
==============================================
Height of tree: 4
==============================================
==============================================
Leaf Nodes:
7 30 60
==============================================
==============================================
Mirror Tree: 60 55 45 30 25 15 12 7
==============================================
==============================================
Original Tree Level Wise:
 45
 12  55
 7  15  60
 25
 30

==============================================
==============================================
Mirror Image of Tree Level Wise:
 45
 55  12
 60  15  7
 25
 30

==============================================
```

Progressive Education Society's

Modern College of Engineering, Pune

MCA Department

A.Y.2022-23

(310902) Data Structure and Algorithm Laboratory

****

Class: FY-MCA Shift /Div: A Batch: F2 Roll Number: 51043

Name: Vanessa Reetu Prashant More Assignment No: 5 Date of Implementation: 23/2/2023

****

Q1. Represent graph using adjacency list/adjacency matrix and perform Depth First Search.

Program:

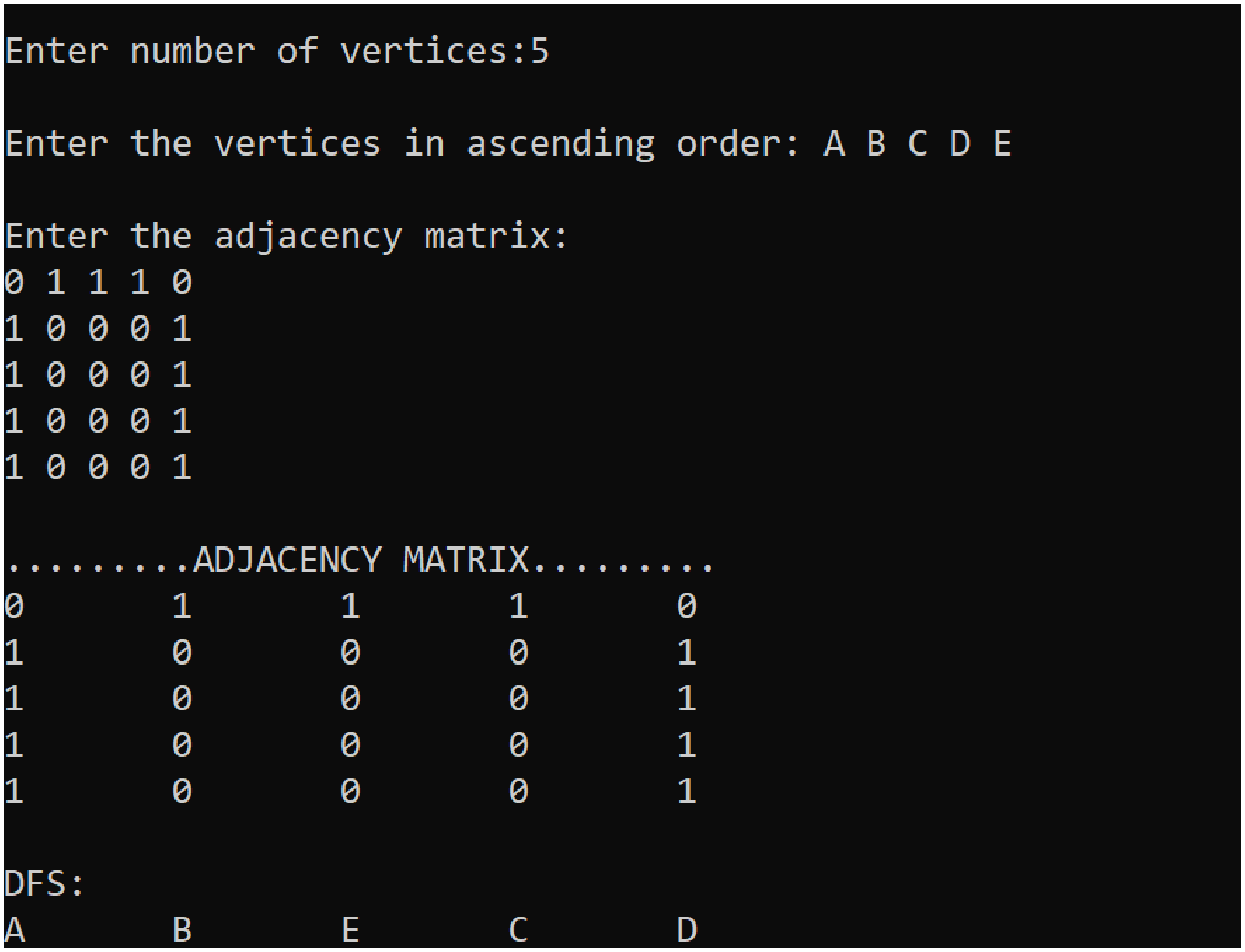
```
#include<stdio.h>
int n,Adj[20][20],visited[10],stack[10];
char vertices[10];
int top=-1;
void push(int v)
{
	if(top==9)
		printf("\nStack is full");
	else
		stack[++top]=v;
}

void pop()
{
	if(top==-1)
		printf("\nStack is empty");
	else
		top--;
}


void DFS(int V)
{
	int vertex = V,f;
	if(visited[vertex]!=1)
	{
		visited[vertex]=1;
		push(vertex);
		printf("%c\t",vertices[vertex]);
		f=0;
		for(int i=0;i<n;i++)
		{
			if(Adj[vertex][i]==1)
			{
				DFS(i);
				f=1;
			}
		}
```

```
                if(f==0)
                        pop();

        }
}


void print_Adj()
{
        int i,j;
                printf("\n.........ADJACENCY MATRIX.........\n");

        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                        printf("%d\t",Adj[i][j]);
                printf("\n");
        }
}


void main()
{
        printf("\nEnter number of vertices:");
        scanf("%d",&n);
        printf("\nEnter the vertices in ascending order: ");
        for(int i=0;i<n;i++)
        {
                scanf("%s",&vertices[i]);
        }
        printf("\nEnter the adjacency matrix:");
        for(int i=0;i<n;i++)
        {
                visited[i]=0;
                stack[i]=0;
                for(int j=0;j<n;j++)
                        scanf("%d",&Adj[i][j]);
        }

        print_Adj();

        printf("\nDFS:\n");
        DFS(0);
}
```

Output:

```
Enter number of vertices:5

Enter the vertices in ascending order: A B C D E

Enter the adjacency matrix:
0 1 1 1 0
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1

.........ADJACENCY MATRIX.........
0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1

DFS:
A	B	E	C	D
```

In Editor:



```
Enter number of vertices:5

Enter the vertices in ascending order: A B C D E

Enter the adjacency matrix:
0 1 1 1 0
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1

.........ADJACENCY MATRIX.........
0       1       1       1       0
1       0       0       0       1
1       0       0       0       1
1       0       0       0       1
1       0       0       0       1

DFS:
A       B       E       C       D
```

Q2. Represent graph using adjacency list/adjacency matrix and perform Breadth First Search.

Program:

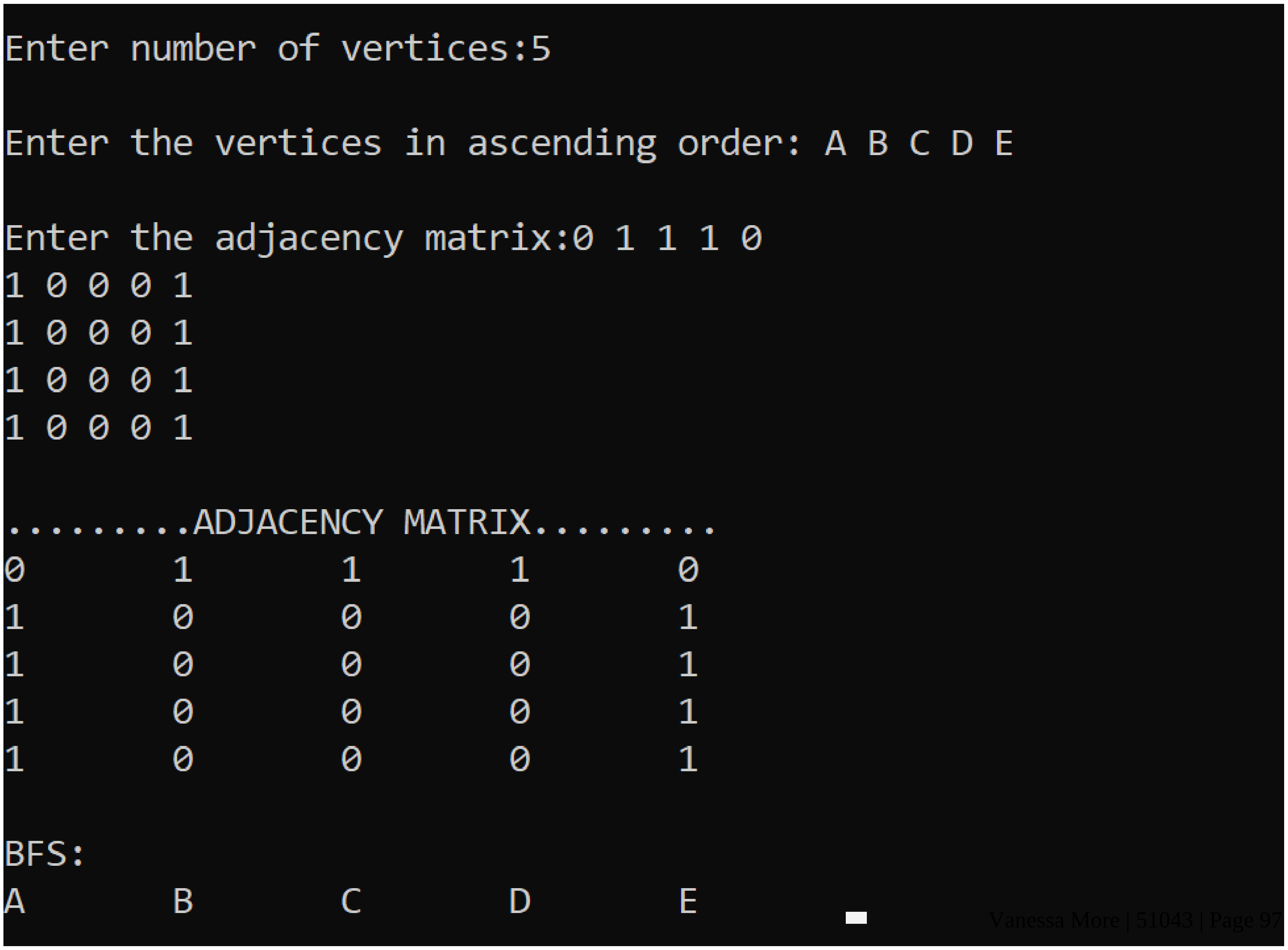
```
#include<stdio.h>
#include<conio.h>
int n,Adj[20][20],visited[10],queue[10];
char vertices[10];
int front=0,rear=-1;
int vc =0;
void enqueue(int v)
{
        if (rear==9)
        {
                printf("\nQueue is full");
        }


        else
        {
                queue[++rear]=v;
                vc++;
        }

}

int dequeue()
{
        /*if(front==rear)
        {
                return -1;
        }

        else   */
        {
                vc--;
                return(queue[front++]);

        }
}

void BFS(int V)
{
        int vertex = V,f=0,i;

                enqueue(vertex);
                printf("%c\t",vertices[vertex]);
                visited[vertex] =1;
                while(vc!=0)
                {
                        int tempvertex = dequeue();
                        if(visited[tempvertex]==1)
                        {
```

```
                    for( i=0;i<n;i++)
                        {
                            if(Adj[tempvertex][i]==1)
                            {
                                if(!visited[i])
                                {
                                    visited[i]=1;
                                    printf("%c\t",vertices[i]);
                                    enqueue(i);
                                }
                            }
                        }
                    }
                }


        }

        void print_Adj()
        {
            int i,j;
            printf("\n.........ADJACENCY MATRIX.........\n");
            for(i=0;i<n;i++)
            {
                for(j=0;j<n;j++)
                    printf("%d\t",Adj[i][j]);
                printf("\n");
            }
        }

        int main()
        {
            int i,j;
            //clrscr();
            printf("\nEnter number of vertices:");
            scanf("%d",&n);
            printf("\nEnter the vertices in ascending order: ");
            for( i=0;i<n;i++)
            {
                scanf("%s",&vertices[i]);
            }
            printf("\nEnter the adjacency matrix:");
            for( i=0;i<n;i++)
            {
                visited[i]=0;
                queue[i]=0;
                for( j=0;j<n;j++)
                    scanf("%d",&Adj[i][j]);
            }

            print_Adj();

            printf("\nBFS:\n");

                BFS(0);
                getch();
                return 0;
        }
```

Output:

```
Enter number of vertices:5

Enter the vertices in ascending order: A B C D E

Enter the adjacency matrix:0 1 1 1 0
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1

.........ADJACENCY MATRIX.........
0       1       1       1       0
1       0       0       0       1
1       0       0       0       1
1       0       0       0       1
1       0       0       0       1

BFS:
A       B       C       D       E
```

In Editor:



```
Enter number of vertices:5

Enter the vertices in ascending order: A B C D E

Enter the adjacency matrix:0 1 1 1 0
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1

.........ADJACENCY MATRIX.........
0       1       1       1       0
1       0       0       0       1
1       0       0       0       1
1       0       0       0       1
1       0       0       0       1

BFS:
A       B       C       D       E
```

Q3. Implement minimum cost spanning tree algorithm.

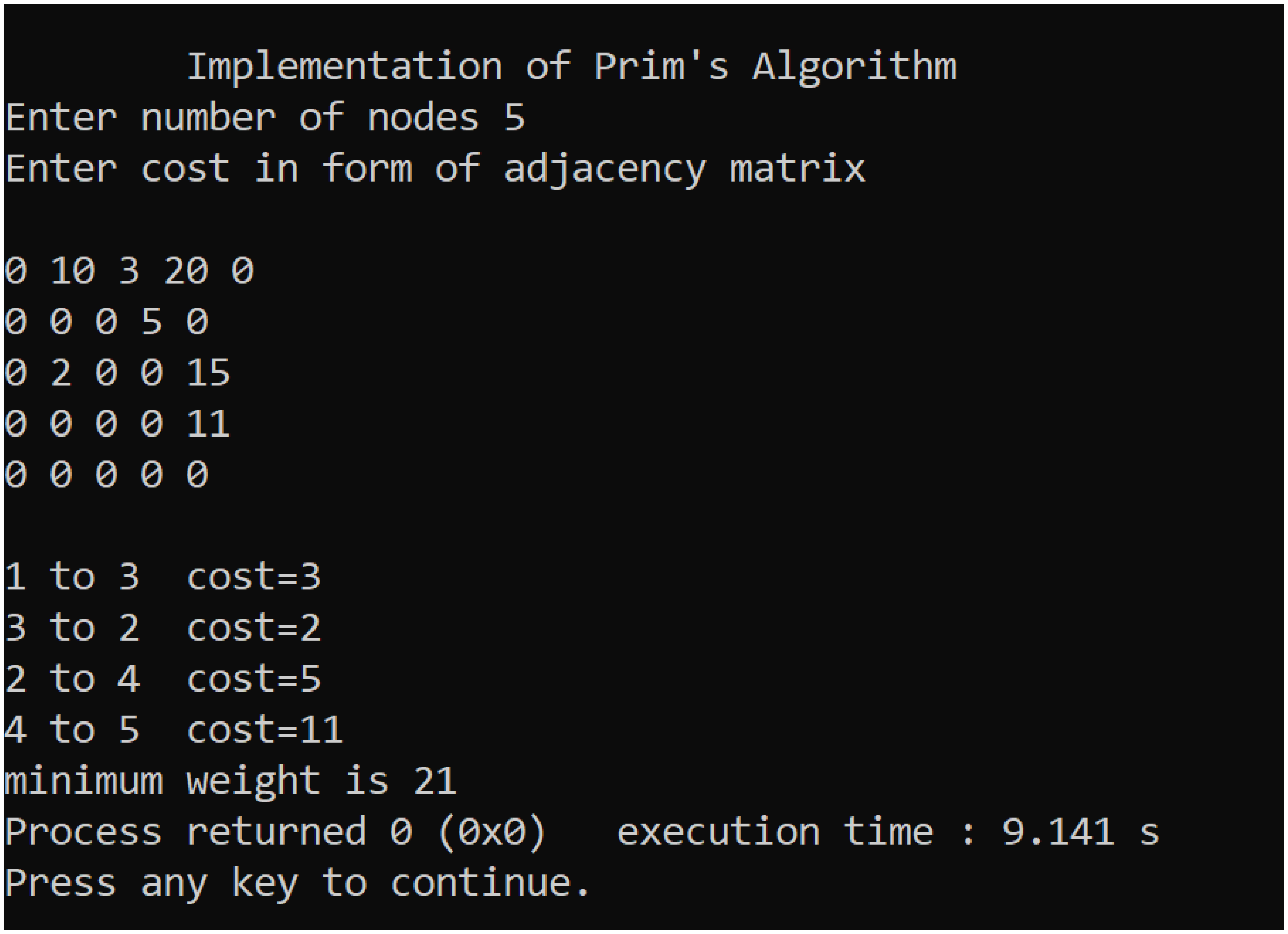
Program 1:

```
#include<stdio.h>
int main()
{
    int cost[10][10],visited[10]={0},i,j,n,no_e=1,min,a,b,min_cost=0;
    printf("\n\tImplementation of Prim's Algorithm\n");
    printf("Enter number of nodes ");
    scanf("%d",&n);
    printf("Enter cost in form of adjacency matrix\n");

    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);

            if(cost[i][j]==0)
              cost[i][j]=1000;
        }
    }


    visited[1]=1; // visited first node
    while(no_e<n)
    {
        min=1000;

        for(i=1;i<=n;i++)
        {
            for(j=1;j<=n;j++)
            {
                if(cost[i][j]<min)
                {
                    if(visited[i]!=0)
                    {
                        min=cost[i][j];
                        a=i;
                        b=j;
                    }
                }
            }
        }

        if(visited[b]==0)
        {
            printf("\n%d to %d  cost=%d",a,b,min);
            min_cost=min_cost+min;
            no_e++;
        }
        visited[b]=1;

        cost[a][b]=cost[b][a]=1000;
    }
    printf("\nminimum weight is %d",min_cost);
    return 0;
}
```

Output:

```
        Implementation of Prim's Algorithm
Enter number of nodes 5
Enter cost in form of adjacency matrix

0 10 3 20 0
0 0 0 5 0
0 2 0 0 15
0 0 0 0 11
0 0 0 0 0

1 to 3  cost=3
3 to 2  cost=2
2 to 4  cost=5
4 to 5  cost=11
minimum weight is 21
```

In Editor:



```
        Implementation of Prim's Algorithm
Enter number of nodes 5
Enter cost in form of adjacency matrix

0 10 3 20 0
0 0 0 5 0
0 2 0 0 15
0 0 0 0 11
0 0 0 0 0

1 to 3  cost=3
3 to 2  cost=2
2 to 4  cost=5
4 to 5  cost=11
minimum weight is 21
Process returned 0 (0x0)   execution time : 9.141 s
Press any key to continue.
```

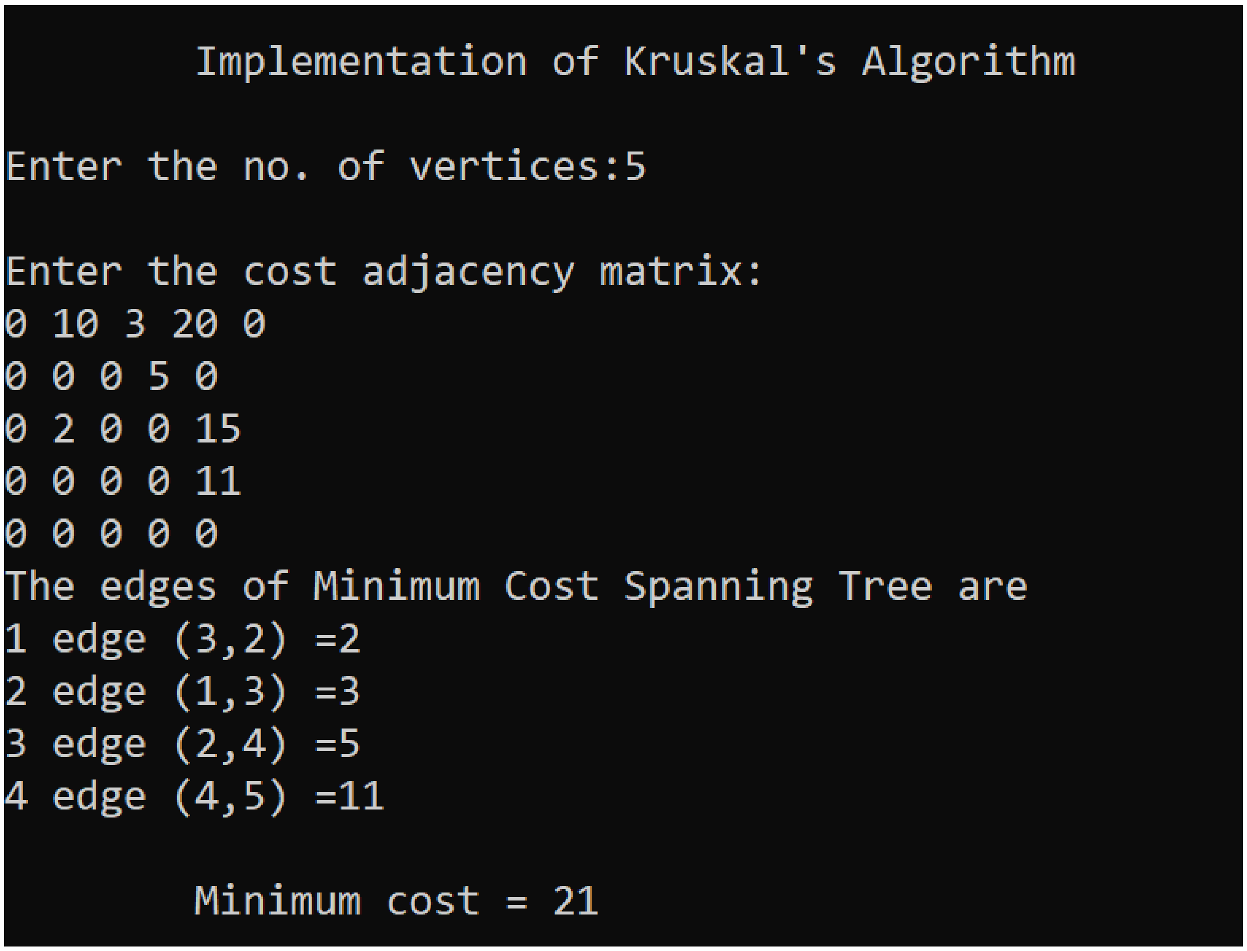
Program 2:

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

int i, j, k, a, b, u, v, n, ne = 1;
int min, mincost = 0, cost[9][9], parent[9];
int find(int);
int uni(int, int);
void main() {
  printf("\n\tImplementation of Kruskal's Algorithm\n");
  printf("\nEnter the no. of vertices:");
  scanf("%d", &n);
  printf("\nEnter the cost adjacency matrix:\n");
  for (i = 1; i <= n; i++) {
    for (j = 1; j <= n; j++) {
      scanf("%d", &cost[i][j]);
      if (cost[i][j] == 0)
        cost[i][j] = 999;
    }
  }
  printf("The edges of Minimum Cost Spanning Tree are\n");
  while (ne < n) {
    for (i = 1, min = 999; i<= n; i++) {
      for (j = 1; j <= n; j++) {
        if (cost[i][j] < min) {
          min = cost[i][j];
          a = u = i;
          b = v = j;
        }
      }
    }
    u = find(u);
    v = find(v);
    if (uni(u, v)) {
      printf("%d edge (%d,%d) =%d\n", ne++, a, b, min);
      mincost += min;
    }
    cost[a][b] = cost[b][a] = 999;
  }
  printf("\n\tMinimum cost = %d\n", mincost);
  getch();
}
int find(int i) {
  while (parent[i])
    i = parent[i];
  return i;
}
int uni(int i, int j) {
  if (i != j) {
    parent[j] = i;
    return 1;
  }
  return 0;
}
```

Output:

```
        Implementation of Kruskal's Algorithm

Enter the no. of vertices:5

Enter the cost adjacency matrix:

0 10 3 20 0
0 0 0 5 0
0 2 0 0 15
0 0 0 0 11
0 0 0 0 0
The edges of Minimum Cost Spanning Tree are
1 edge (3,2) =2
2 edge (1,3) =3
3 edge (2,4) =5
4 edge (4,5) =11

        Minimum cost = 21
```

In Editor:



```
        Implementation of Kruskal's Algorithm

Enter the no. of vertices:5

Enter the cost adjacency matrix:
0 10 3 20 0
0 0 0 5 0
0 2 0 0 15
0 0 0 0 11
0 0 0 0 0
The edges of Minimum Cost Spanning Tree are
1 edge (3,2) =2
2 edge (1,3) =3
3 edge (2,4) =5
4 edge (4,5) =11

        Minimum cost = 21
```

Q4. Implement shortest path algorithm.

Program:

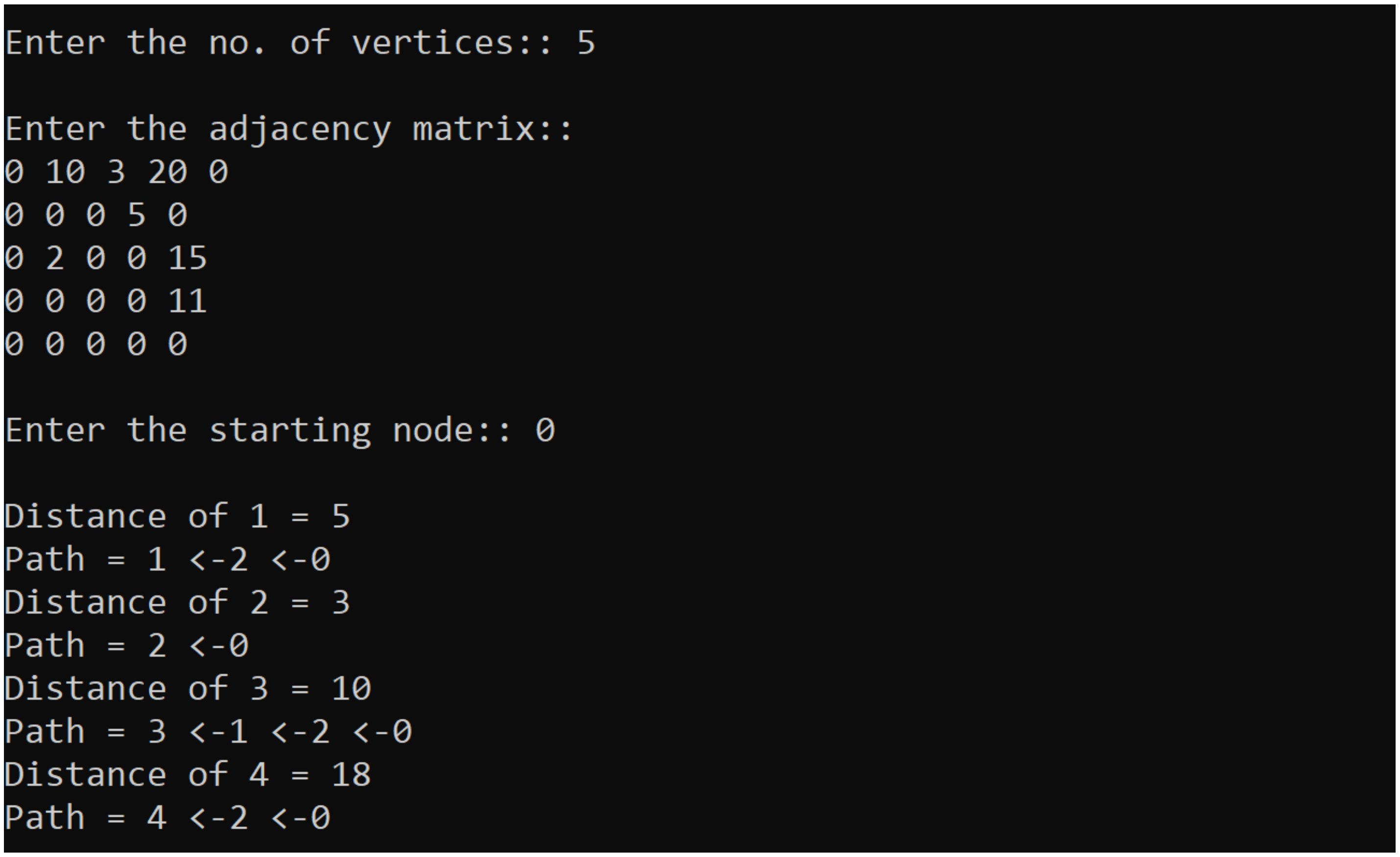
```
#include<stdio.h>
#include<conio.h>
#define INFINITY 9999
#define MAX 10

void dijikstra(int G[MAX][MAX], int n, int startnode);

int main(){
	int G[MAX][MAX], i, j, n, u;

	printf("\nEnter the no. of vertices:: ");
	scanf("%d", &n);
	printf("\nEnter the adjacency matrix::\n");
	for(i=0;i < n;i++)
		for(j=0;j < n;j++)
			scanf("%d", &G[i][j]);
	printf("\nEnter the starting node:: ");
	scanf("%d", &u);
	dijikstra(G,n,u);
	getch();
	return 0;
}

void dijikstra(int G[MAX][MAX], int n, int startnode)
{
	int cost[MAX][MAX], distance[MAX], pred[MAX];
	int visited[MAX], count, mindistance, nextnode, i,j;
	for(i=0;i < n;i++)
		for(j=0;j < n;j++)
			if(G[i][j]==0)
				cost[i][j]=INFINITY;
			else
				cost[i][j]=G[i][j];

	for(i=0;i< n;i++)
	{
		distance[i]=cost[startnode][i];
		pred[i]=startnode;
		visited[i]=0;
	}
	distance[startnode]=0;
	visited[startnode]=1;
	count=1;
	while(count < n-1){
		mindistance=INFINITY;
		for(i=0;i < n;i++)
			if(distance[i] < mindistance&&!visited[i])
			{
				mindistance=distance[i];
				nextnode=i;
			}
		visited[nextnode]=1;
```

```
                        for(i=0;i < n;i++)
                                if(!visited[i])
                                        if(mindistance+cost[nextnode][i] < distance[i])
                                        {
                                                distance[i]=mindistance+cost[nextnode][i];
                                                pred[i]=nextnode;
                                        }
                                count++;
                }

                for(i=0;i < n;i++)
                        if(i!=startnode)
                        {
                                printf("\nDistance of %d = %d", i, distance[i]);
                                printf("\nPath = %d", i);
                                j=i;
                                do
                                {
                                        j=pred[j];
                                        printf(" <-%d", j);
                                }
                                while(j!=startnode);

                        }
        }
```

Output:

```
Enter the no. of vertices:: 5

Enter the adjacency matrix::
0 10 3 20 0
0 0 0 5 0
0 2 0 0 15
0 0 0 0 11
0 0 0 0 0

Enter the starting node:: 0

Distance of 1 = 5
Path = 1 <-2 <-0
Distance of 2 = 3
Path = 2 <-0
Distance of 3 = 10
Path = 3 <-1 <-2 <-0
Distance of 4 = 18
Path = 4 <-2 <-0
```

In Editor:



```
Enter the no. of vertices:: 5

Enter the adjacency matrix::
0 10 3 20 0
0 0 0 5 0
0 2 0 0 15
0 0 0 0 11
0 0 0 0 0

Enter the starting node:: 0

Distance of 1 = 5
Path = 1 <-2 <-0
Distance of 2 = 3
Path = 2 <-0
Distance of 3 = 10
Path = 3 <-1 <-2 <-0
Distance of 4 = 18
Path = 4 <-2 <-0
```

****
Class: FY-MCA Shift /Div: A Batch: F2 Roll Number: 51043

Name: Vanessa Reetu Prashant More Assignment No: 6 Date of Implementation: 24/2/2023
****
Q1. Write a program to implement Merge sort method.

Program:

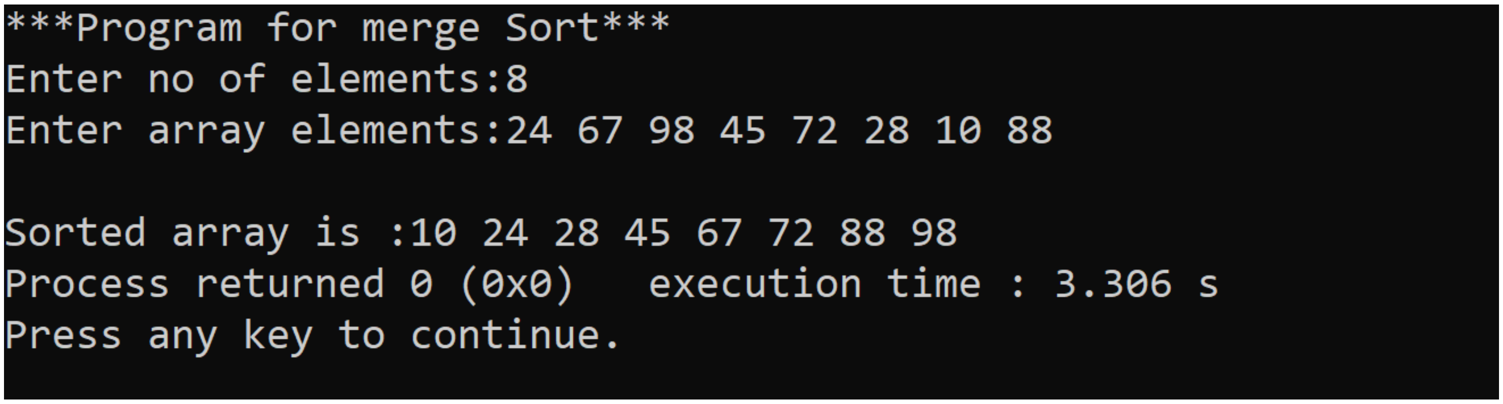
```
#include<stdio.h>

void merge(int a[],int i1,int j1,int i2,int j2)
{
    int temp[50];
    int i,j,k;
    i=i1;
    j=i2;
    k=0;
      while(i<=j1 && j<=j2)        {
         if(a[i]<a[j])
         temp[k++]=a[i++];
         else
         temp[k++]=a[j++];
      }
    while(i<=j1)
      temp[k++]=a[i++];
    while(j<=j2)
      temp[k++]=a[j++];

    for(i=i1,j=0;i<=j2;i++,j++)
      a[i]=temp[j];
}


void mergesort(int a[],int i,int j)
{
int mid;
  if(i<j)
    {
      mid=(i+j)/2;
      mergesort(a,i,mid);
      mergesort(a,mid+1,j);
      merge(a,i,mid,mid+1,j);

    }
}
```

```
int main()
{
    int a[30],n,i;
    printf("***Program for merge Sort***\n");
    printf("Enter no of elements:");
    scanf("%d",&n);
    printf("Enter array elements:");
        for(i=0;i<n;i++)
            scanf("%d",&a[i]);

    mergesort(a,0,n-1);

    printf("\nSorted array is :");
        for(i=0;i<n;i++)
            printf("%d ",a[i]);
    return 0;
}
```

Output:

```
***Program for merge Sort***
Enter no of elements:8
Enter array elements:24 67 98 45 72 28 10 88

Sorted array is :10 24 28 45 67 72 88 98
```

In Editor:



```
***Program for merge Sort***
Enter no of elements:8
Enter array elements:24 67 98 45 72 28 10 88

Sorted array is :10 24 28 45 67 72 88 98
Process returned 0 (0x0)   execution time : 3.306 s
Press any key to continue.
```

Q2. Write a program to implement Bubble sort method.

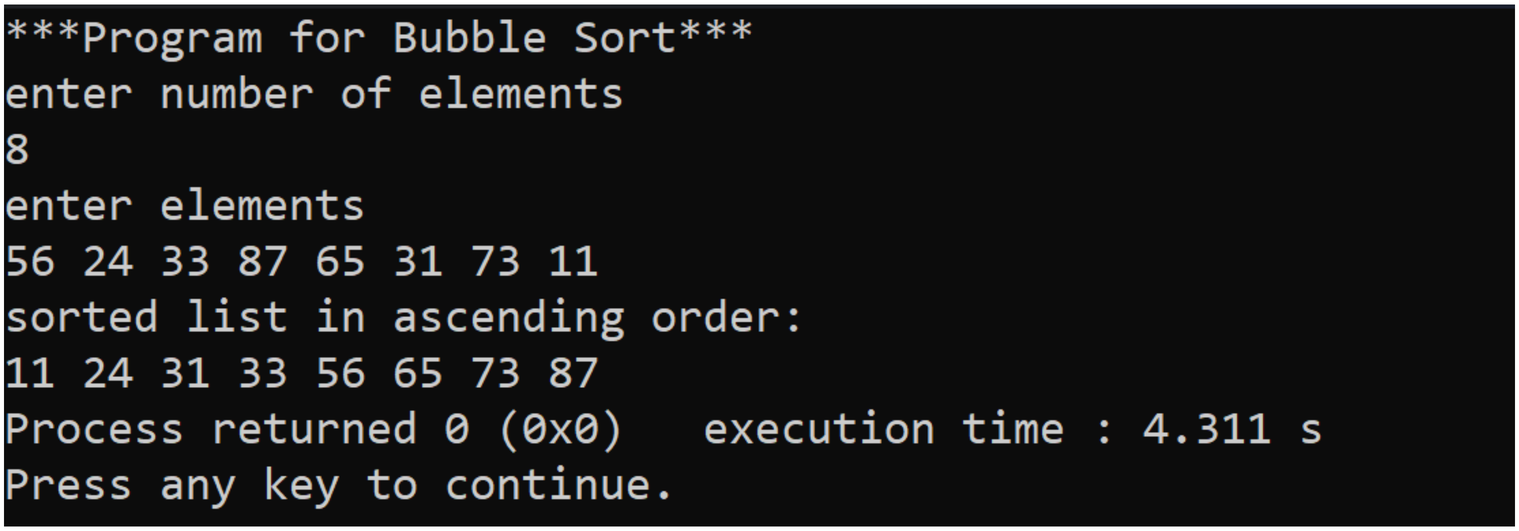
Program:

```
#include<stdio.h>
int main()
{
  int array[100],n,c,d,swap;
  printf("***Program for Bubble Sort***\n");
  printf("enter number of elements \n");
  scanf("%d",&n);
  printf("enter elements \n",n);
  for(c=0;c<n;c++)
    scanf("%d",&array[c]);

  for(c=0;c<n-1;c++)
  {
    for(d=0;d<n-c-1;d++)
    {
      if(array[d]>array[d+1])
      {
        swap=array[d];
        array[d]=array[d+1];
        array[d+1]=swap;

      }
    }

}
printf("sorted list in ascending order:\n");
for(c=0;c<n;c++)

  printf("%d ",array[c]);


return 0;
}
```

Output:

```
***Program for Bubble Sort***
enter number of elements
8
enter elements
56 24 33 87 65 31 73 11
sorted list in ascending order:
11 24 31 33 56 65 73 87
```

In Editor:



```
***Program for Bubble Sort***
enter number of elements
8
enter elements
56 24 33 87 65 31 73 11
sorted list in ascending order:
11 24 31 33 56 65 73 87
Process returned 0 (0x0)   execution time : 4.311 s
Press any key to continue.
```

Q3. Write a program to implement Fibonacci Search.

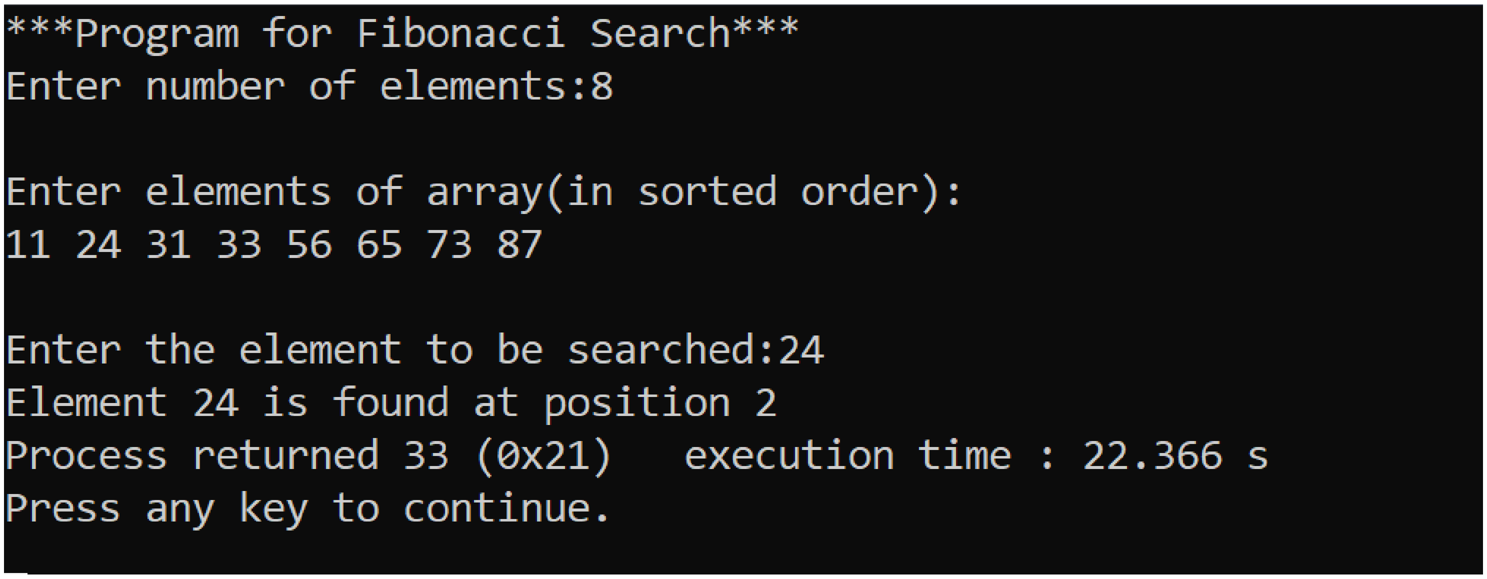
Program:

```
#include<stdio.h>
#include<stdlib.h>
#define min(a,b) a<b?a:b;

void main()
{
    int n,ar[10],x,i,a,b,c,offset;
    printf("***Program for Fibonacci Search***\n");
    printf("Enter number of elements:");
    scanf("%d",&n);
    printf("\nEnter elements of array(in sorted order): ");
    for(i=0;i<n;i++)
            {
        scanf("%d",&ar[i]);
    }
    printf("\nEnter the element to be searched:");
    scanf("%d",&x);
    a=0;
    b=1;
    c=a+b;
    while(c<n)
            {
        a=b;
        b=c;
        c=a+b;
    }
    offset=-1;
    while(c>1)
            {
                    i=min(offset+a,n-1);
        if(ar[i]<x)
                    {
            c=b;
            b=a;
            a=c-b;
            offset=i;
        }
                    else if(ar[i]>x)
                    {
            c=a;
            b=b-a;
            a=c-b;
        }
        else
                    {
            printf("Element %d is found at position %d",x,i+1);
            return;
        }
    }
if(b && ar[offset+1]==x)
         {
      printf("Element %d is found at index %d",x,offset);
      return;
   }
   printf("\nElement not found.");

}
```

Output:

```
***Program for Fibonacci Search***
Enter number of elements:8

Enter elements of array(in sorted order):
11 24 31 33 56 65 73 87

Enter the element to be searched:24
Element 24 is found at position 2
```

In Editor:



```
***Program for Fibonacci Search***
Enter number of elements:8

Enter elements of array(in sorted order):
11 24 31 33 56 65 73 87

Enter the element to be searched:24
Element 24 is found at position 2
Process returned 33 (0x21)   execution time : 22.366 s
Press any key to continue.
```

Q4. Write a program to implement Heap sort method.

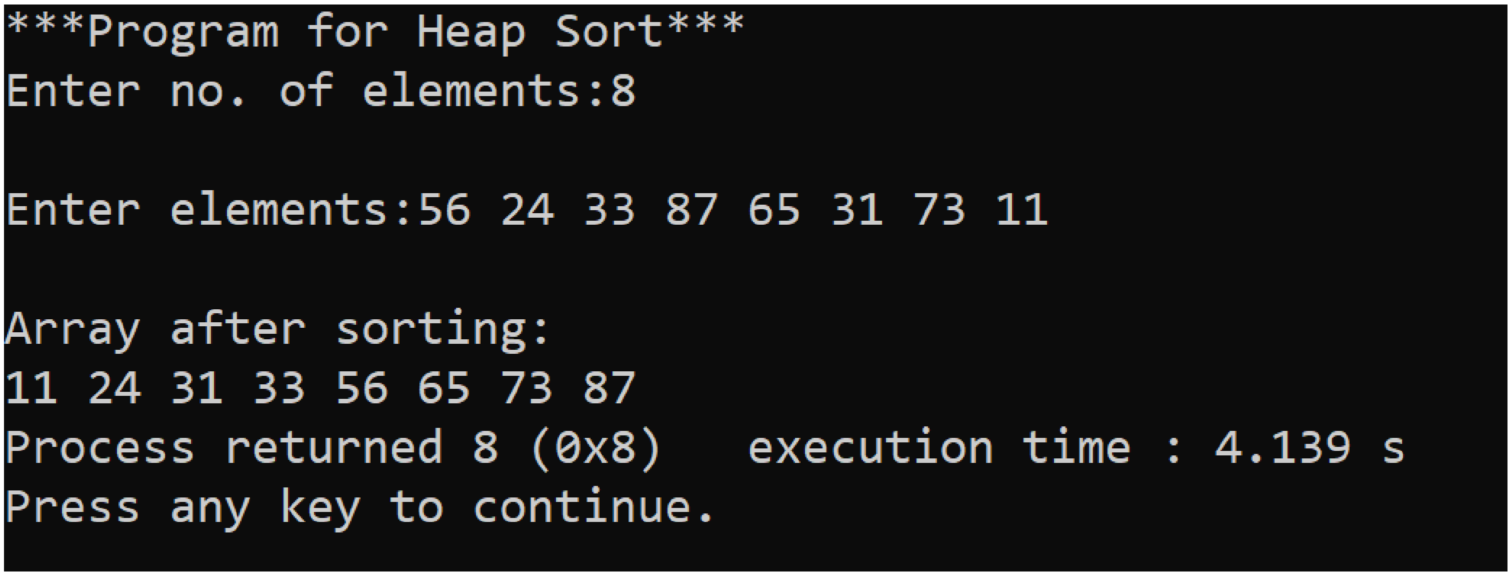
Program:

```
#include<stdio.h>

void down_adjust(int heap[],int i)
{
    int j,temp,n,flag=1;
    n=heap[0];
        while(2*i<=n && flag==1)
            {
                j=2*i;
                if(j+1<=n && heap[j+1] > heap[j])
                    j=j+1;
                if(heap[i] > heap[j])
                    flag=0;
                else
                {
                    temp=heap[i];
                    heap[i]=heap[j];
                    heap[j]=temp;
                    i=j;
                }
            }
}

void create(int heap[])
{
    int i,n;
    n=heap[0]; //no. of elements
    for(i=n/2;i>=1;i--)
    down_adjust(heap,i);
}

void main()
{
    int heap[30],n,i,last,temp;
    printf("***Program for Heap Sort***\n");
    printf("Enter no. of elements:");
    scanf("%d",&n);
    printf("\nEnter elements:");
        for(i=1;i<=n;i++)
        scanf("%d",&heap[i]);
        //create a heap
        heap[0]=n;
        create(heap);
while(heap[0] > 1)
      {
         //swap heap[1] and heap[last]
         last=heap[0];
         temp=heap[1];
         heap[1]=heap[last];
         heap[last]=temp;
         heap[0]--;
         down_adjust(heap,1);
      }

   //print sorted data
   printf("\nArray after sorting:\n");

   for(i=1;i<=n;i++)
      printf("%d ",heap[i]);
}
```

Output:

```
***Program for Heap Sort***
Enter no. of elements:8

Enter elements:56 24 33 87 65 31 73 11

Array after sorting:
11 24 31 33 56 65 73 87
```

In Editor:



```
***Program for Heap Sort***
Enter no. of elements:8

Enter elements:56 24 33 87 65 31 73 11

Array after sorting:
11 24 31 33 56 65 73 87
Process returned 8 (0x8)   execution time : 4.139 s
Press any key to continue.
```
