DEPARTMENT VISION

TO BE A CENTER OF EXCELLENCE FOR NURTURING THE YOUNG MINDS TO BECOME INNOVATIVE COMPUTING PROFESSIONALS FOR THE EMPOWERMENT OF SOCIETY.

DEPARTMENT MISSION

1. TO OFFER A SOLID FOUNDATION IN COMPUTING AND TECHNOLOGY FOR CRAFTING COMPETENT PROFESSIONALS.

2. TO PROMOTE INNOVATIVE AND ENTREPRENEURIAL SKILLS OF STUDENTS BY EXPOSING THEM TO THE FOREFRONT OF DEVELOPMENTS IN THE FIELD OF COMPUTING.

3. TO INCULCATE STRONG ETHICAL VALUES IN THE YOUNG MINDS TO WORK WITH COMMITMENT FOR THE PROGRESS OF THE NATION.

COURSE OUTCOMES

At the end of the course, the student should be able to

CO1	Design an algorithm for a computational task and calculate the time/space complexities of that algorithm (Cognitive Knowledge Level: Apply)
CO2	Identify the suitable data structure (array or linkedlist) to represent a data item required to be processedto solve a given computational problem and write an algorithm to find the solution of the computational problem (Cognitive Knowledge Level: Apply)
CO3	Write an algorithm to find the solution of a computational problem by selecting an appropriate data structure (binary tree/graph) to represent a data item to be processed (Cognitive Knowledge Level: Apply)
CO4	Store a given dataset using an appropriate Hash Function to enable efficient access of data in the given set (Cognitive Knowledge Level: Apply)
CO5	Select appropriate sorting algorithms to be used in specific circumstances (Cognitive Knowledge Level: Analyze)
CO6	Design and implement Data Structures for solving real world problems efficiently (Cognitive Knowledge Level: Apply)

INDEX

Experiment No: 01

POLYNOMIAL ADDITION

PROBLEM DEFINITION:

Write a program to implement Polynomial Addition.

ALGORITHM:

```
Step 1: Start
Step 2: Read coefficient &exponent of 2 polynomials
Step 3: Compare the exponents from 1st node
Step 4: If (poly1->coeff>poly2->coeff)
        Step 4.1: Poly->coeff=poly1->coeff
        Step 4.2: Poly->exp=poly1->exp
        Step 4.3: Poly1=poly1->link
        Step 4.4: Poly->link = getnode (node)
        Step 4.5: Poly=poly->link
        Step 4.6: Exit
Step 5: Elseif (poly->exp<poly2->exp)
        Step 5.1: Poly->coeff=poly2->coeff
        Step 5.2: Poly->exp=poly2->exp
        Step 5.3:Poly2=poly2->link
        Step 5.4: Poly->link = getnode (node)
        Step 5.5: Poly=poly->link
        Step 5.6: Exit
Step 6: Else
        Step 6.1: Poly->coeff=poly2->coeff+poly1->coeff
        Step 6.2: Poly->exp=poly1->exp
        Step 6.3: Poly2=poly2->link
        Step 6.4: Poly1=poly1->link
        Step 6.5: Poly->link = getnode (node)
        Step 6.6: Poly=poly->link
        Step 6.7: Exit
Step 7: If (poly1->link!=Null)
        Step 7.1: While (poly1->link!=Null)
                Step 7.1.1: Poly->coeff=poly1->coeff
                Step 7.1.2: Poly->exp=poly1->exp
                Step 7.1.3: Poly1=poly1->link
                Step 7.1.4: Poly->link=getnode (node)
                Step 7.1.5: Poly=polylink
        Step 7.2: End while
Step 8: Else if
        Step 8.1: While (poly2link!=Null)
                Step 8.1.1: Poly->coeff=poly2->coeff
                Step 8.1.2: Poly->exp=poly2->exp
                Step 8.1.3: Poly2=poly2->link
                Step 8.1.4: Poly->link = getnode (node)
                Step 8.1.5: Poly=poly->link
        Step 8.2: End while
Step 9: End if
Step10:

Stop
```

PROGRAM DEVELOPMENT:

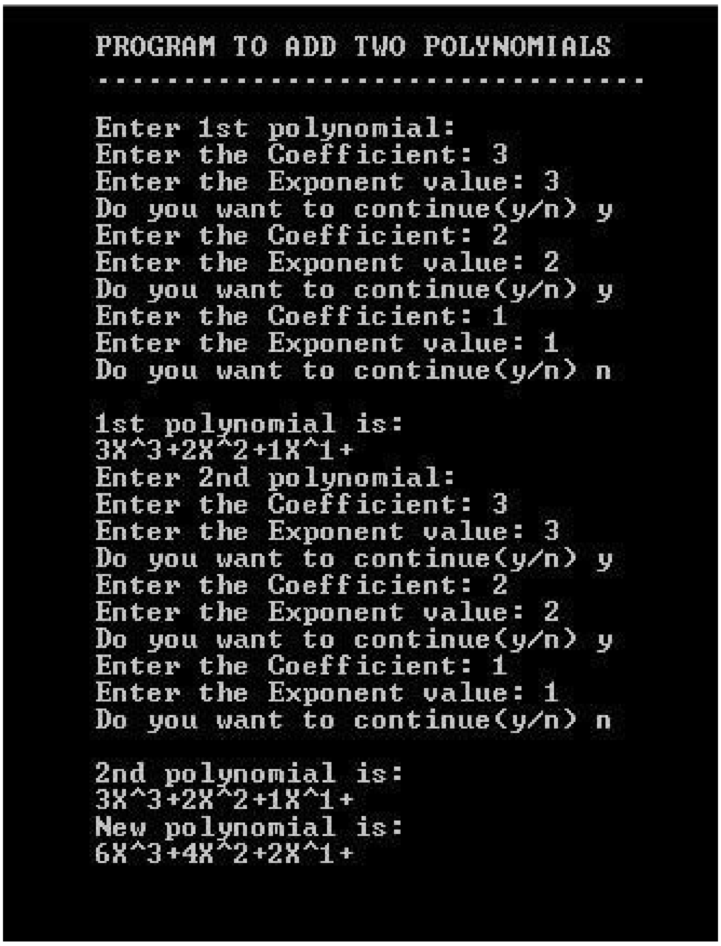
```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
typedef struct node
{ int coef
f; int exp;
struct node*link;
}NODE;
NODE
*poly1=NULL,*poly2=NULL,*poly=NULL;
void create(NODE*);
void show(NODE*);
void polyadd(NODE*,NODE*,NODE*);
void main()
{
clrscr();
printf("\n\t\tPROGRAM TO ADD TWO POLYNOMIALS\n");
printf("\t\t                    \n");
poly=(NODE*)malloc(sizeof(NODE));
poly1=(NODE*)malloc(sizeof(NODE));
poly2=(NODE*)malloc(sizeof(NODE));
printf("\n\t\tEnter 1st polynomial: ");
create(poly1);
printf("\n\t\t1st polynomial is: ");
show(poly1);
printf("\n\t\tEnter 2nd
polynomial: "); create(poly2);
```

```
printf("\n\t\t2nd polynomial is:");
show(poly2);
polyadd(poly1,poly2,poly);
 printf("\n\t\tNew polynomial is:");
show(poly);
getch();
}
void create(NODE*ptr)
{ char c;
 printf("\n");
 do
{ printf("\t\tEnter the Coefficient: ");
scanf("%d",&ptr->coeff)
; printf("\t\tEnter the Exponent value:
"); scanf("%d",&ptr->exp);
 ptr->link=(NODE*)malloc(sizeof(NODE));

ptr=ptr->link;
 ptr->link=NULL;
printf("\t\tDo you want to continue(y/n) ");
scanf(" %c",&c);
}
while(c=='y'||c=='Y');
}
void show(NODE*ptr)
{printf("\n\t\ t");
while(ptr->link!=NULL)
{
if(ptr->exp==0)
printf("%d",ptr->coeff);
else
printf("%dX^%d+",ptr->coeff,ptr->exp);
ptr=ptr->link;
}
}
void
polyadd(NODE*ptr1,NODE*ptr2,NODE*ptr)
{
while(ptr1->link!=NULL&&ptr2->link!=NULL)
```

```
{
if(ptr1->exp>ptr2->exp)
{
ptr->coeff=ptr1->coeff;
 ptr->exp=ptr1->exp;
 ptr1=ptr1->link;
ptr->link=(NODE*)malloc(sizeof(NODE));
 ptr=ptr->link;
ptr->link=NULL;
}
else if(ptr1->exp<ptr2->exp)
{
ptr->coeff=ptr2->coeff;
 ptr->exp=ptr2->exp;
 ptr2=ptr2->link;
ptr->link=(NODE*)malloc(sizeof(NODE));
ptr=ptr->link; ptr->link=NULL;
}
else

{
ptr->coeff=ptr1->coeff+ptr2->coeff;
ptr->exp=ptr1->exp;
 ptr1=ptr1->link;
ptr2=ptr2->link;
ptr->link=(NODE*)malloc(sizeof(NODE));
ptr=ptr->link;
 ptr->link=NULL;
}
}
if(ptr1->link!=NULL)
{
while(ptr1->link!=NULL)
{
ptr->coeff=ptr1->coeff;
 ptr->exp=ptr1->exp;
 ptr1=ptr1->link;
 ptr->link=(NODE*)malloc(sizeof(NODE));
ptr=ptr->link;
ptr->link=NULL;
}
}
else if(ptr2->link!=NULL)
```

```
{
while(ptr2->link! =NULL)
{
ptr->coeff=ptr2->coeff;
ptr->exp=ptr2->exp;
ptr2=ptr2->link;
ptr->link=(NODE*)malloc(sizeof(NODE));
 ptr=ptr->link;
 ptr->link=NULL;
}
}
}
```

OUTPUT



```
PROGRAM TO ADD TWO POLYNOMIALS
--------------------------------

Enter 1st polynomial:
Enter the Coefficient: 3
Enter the Exponent value: 3
Do you want to continue(y/n) y
Enter the Coefficient: 2
Enter the Exponent value: 2
Do you want to continue(y/n) y
Enter the Coefficient: 1
Enter the Exponent value: 1
Do you want to continue(y/n) n

1st polynomial is:
3X^3+2X^2+1X^1+
Enter 2nd polynomial:
Enter the Coefficient: 3
Enter the Exponent value: 3
Do you want to continue(y/n) y
Enter the Coefficient: 2
Enter the Exponent value: 2
Do you want to continue(y/n) y
Enter the Coefficient: 1
Enter the Exponent value: 1
Do you want to continue(y/n) n

2nd polynomial is:
3X^3+2X^2+1X^1+
New polynomial is:
6X^3+4X^2+2X^1+
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 02

SPARSE MATRIX

PROBLEM DEFINITION:

To develop a program that implements sparse matrix using array, find its transpose and add the spares matrices.

ALGORITHM:

//For Representation

Step 1 : Start

Step 2 : Declare and initialize the values of a 4*5 sparse matrix.

Note: If no: of zeroes is greater than row*col/2, then the matrix is sparse , else not.

Step 3 : Declare a variable, COUNT to count the number of non zero elements and initialize it as 0.

Step 4 : Check every i, j th element, whether it is non zero, if non zero, increment above variable by 1.

Step 5 : Declare a resultant sparse matrix with 3 rows and COUNT (no: of non zero elements) columns

Step 6 : Iterate through every i, jth element, On every non zero element corresponding row value, column value, and the i,jth element is saved to resultant matrix

//For Transpose

(1) for each row i
    a. take element and store it in element of the transpose.
(2) difficulty: where to put (0, 0, 15) ====> (0, 0, 15) (0, 3, 22) ====> (3, 0, 22) (0, 5, -15) ====> (5, 0, -15) (1, 1, 11) ====> (1, 1, 11) Move elements down very often. (2) For all elements in column j, place element in element <j,

//For Addition

```
Step1: Initialize i = 1, j=1, k=1

Step2:while(i <= A[0].val && j<= B[0].val)

        if(A[i].row == B[j].row && A[i].col == B[j].col)

                C[k].row = A[i].row

                C[k].col = A[i].col

                C[k].val = A[i].val + B[j].val

                i++, j++, k++

    if(A[i].row == B[j].row && A[i].col < B[j].col)
```

```
                C[k].row = A[i].row
                C[k].col = A[i].col
                C[k].val = A[i].val
                i++, k++
        if(A[i].row == B[j].row && A[i].col > B[j].col)
                C[k] = B[j]
                j++, k++
        if(A[i].row < B[j].row )
                C[k] = A[i] ′
                i++, k++
        if(A[i].row > B[j].row )
                C[k] = B[j]
                j++, k++
        // end while ′
Step3: while(i<=A[0].val)
        C[k] = A[i] • i++,
        k++ ′
Step 4: while(j <=B[0].val)
        C[k] = B[j]
        j++, k++
Step5:C[0].row = A[0].row, C[0].col = A[0].col, C[0].val = k
```

PROGRAM DEVELOPMENT:

```
//Program to represent Sparse matrix
#include <stdio.h>
#define MAX_VAL 101
struct triple
{
        int row;
        int col;
        int val;
} sparse[MAX_VAL];

void tripletrep(int arr[1000][1000], int n, int m);
void print(int k)
{
        printf("The Triplet representation is\n");
        int i, j;
        for (i = 0; i < k; i++)
        {
                printf("%d %d %d \n", sparse[i].row, sparse[i].col, sparse[i].val);
        }
}

int main()
{
        int i, j;
        printf("Enter the Order of the Sparse array: ");
        int n, m;
        scanf("%d %d", &n, &m);
        int arr[1000][1000];
        printf("Enter the elements: ");
        for (i = 0; i < n; i++)
        {
                for (j = 0; j < m; j++)
                        scanf("%d", &arr[i][j]);
        }
        tripletrep(arr, n, m);
}
```

```
void tripletrep(int arr[1000][1000], int n, int m)
{
        int i, j, k = 1;
        for (i = 0; i < n; i++)
                for (j = 0; j < m; j++)
                {
                        if (arr[i][j] != 0)
                        {
                                sparse[k].row = i;
                                sparse[k].col = j;
                                sparse[k].val = arr[i][j];
                                k++;
                        }
                }
        sparse[0].row = i;
        sparse[0].col = j;
        sparse[0].val = k - 1;
        print(k);
}
//Program to find transpose
#include <stdio.h>
#define MAX_VAL 101
struct triple
{
        int row;
        int col;
        int val;
} sparse[MAX_VAL];
```

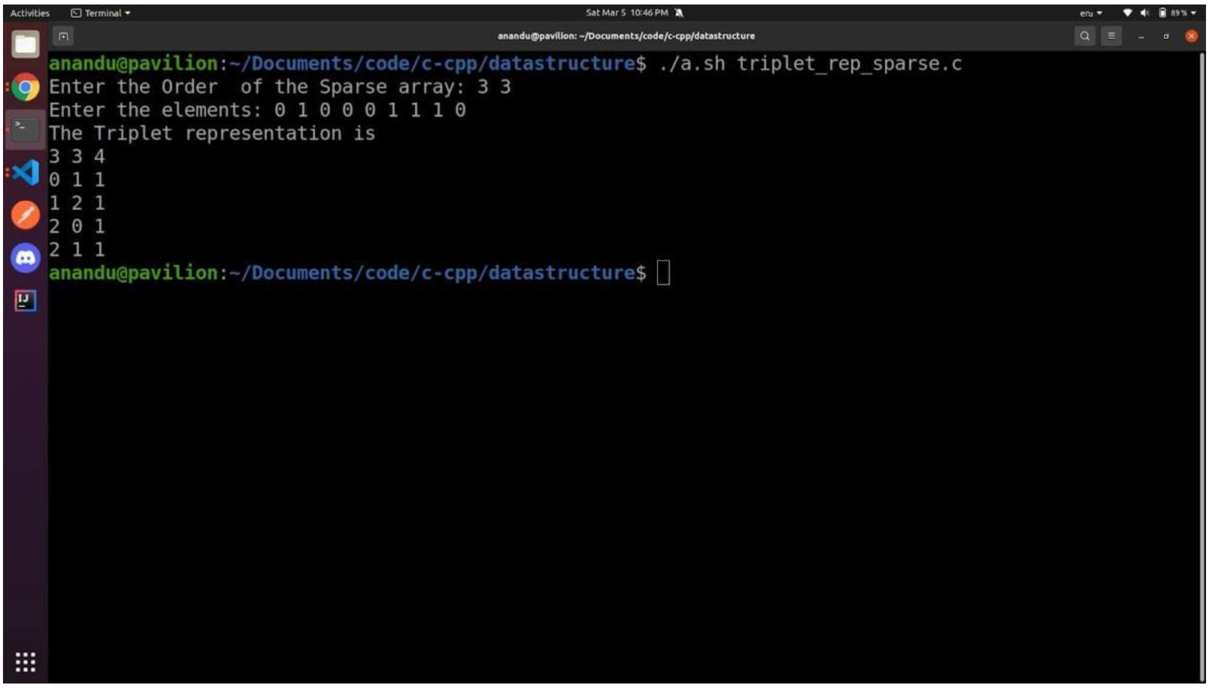
```
void tripletrep(int arr[1000][1000], int n, int m);
void print(int k)
{
        printf("The Triplet representation is\n");
        int i, j;
        for (i = 0; i < k; i++)
        {
                printf("%d %d %d \n", sparse[i].row, sparse[i].col, sparse[i].val);
        }
}

int main()
{
        int i, j;
        printf("Enter the Order of the Sparse array: ");
        int n, m;
        scanf("%d %d", &n, &m);
        int arr[1000][1000];
        printf("Enter the elements: ");
        for (i = 0; i < n; i++)
        {
                for (j = 0; j < m; j++)
                        scanf("%d", &arr[i][j]);
        }
        tripletrep(arr, n, m);
}
void tripletrep(int arr[1000][1000], int n, int m)
{
        int i, j, k = 1;
        for (i = 0; i < n; i++)
                for (j = 0; j < m; j++)
                {
                        if (arr[i][j] != 0)
                        {
                                sparse[k].row = i;
                                sparse[k].col = j;
                                sparse[k].val = arr[i][j];
                                k++;
                        }
                }
        sparse[0].row = i;
        sparse[0].col = j;
        sparse[0].val = k - 1;
        print(k);
}

//Program to find sum
#include <stdio.h>
#define MAX_VAL 101
struct triple
{
        int row;
        int col;
        int val;
} a[MAX_VAL], b[MAX_VAL], c[MAX_VAL];

void print(int n)
{
        int i, j;
        for (i = 0; i <= n; i++)
        {
                printf("%d %d %d \n", c[i].row, c[i].col, c[i].val);
        }
}
void add(int n, int m)
{
        int i, j, k;
        if (a[0].row == b[0].row && a[0].col == b[0].col)
        {
                c[0].row = a[0].row;
                c[0].col = a[0].col;
                i = j = k = 1;

                while (i <= n && j <= m)
                {
                        if (a[i].row == b[j].row && a[i].col == b[j].col)
                        {
                                c[k].row = a[i].row;
                                c[k].col = a[i].col;
                                c[k].val = a[i].val + b[j].val;
                                k++;
                                i++;
                                j++;
                        }
                        else if (a[i].row == b[j].row)
                        {
```

```
                        if (a[i].col < b[j].col)
                        {
                                    c[k].row = a[i].row;
                                    c[k].col = a[i].col;
                                    c[k].val=a[i].val;
                                    k++;

                                    i++;
                        }
                        else
                        {
                                    c[k].row = b[j].row;
                                    c[k].col = b[j].col;
                                    c[k].val
                        }           =
                            }       b[j].val;
                                    k++;
                                    j++;
                    else if (a[i].row < b[i].row)
                    {
}
                    else
                    {
  c[k].row = a[i].row;
  c[k].col = a[i].col;
  c[k].val = a[i].val;
  k++;i++;
  c[k].row = b[j].row;
  c[k].col = b[j].col;
  c[k].val = b[j].val;
  k++; j++;
```

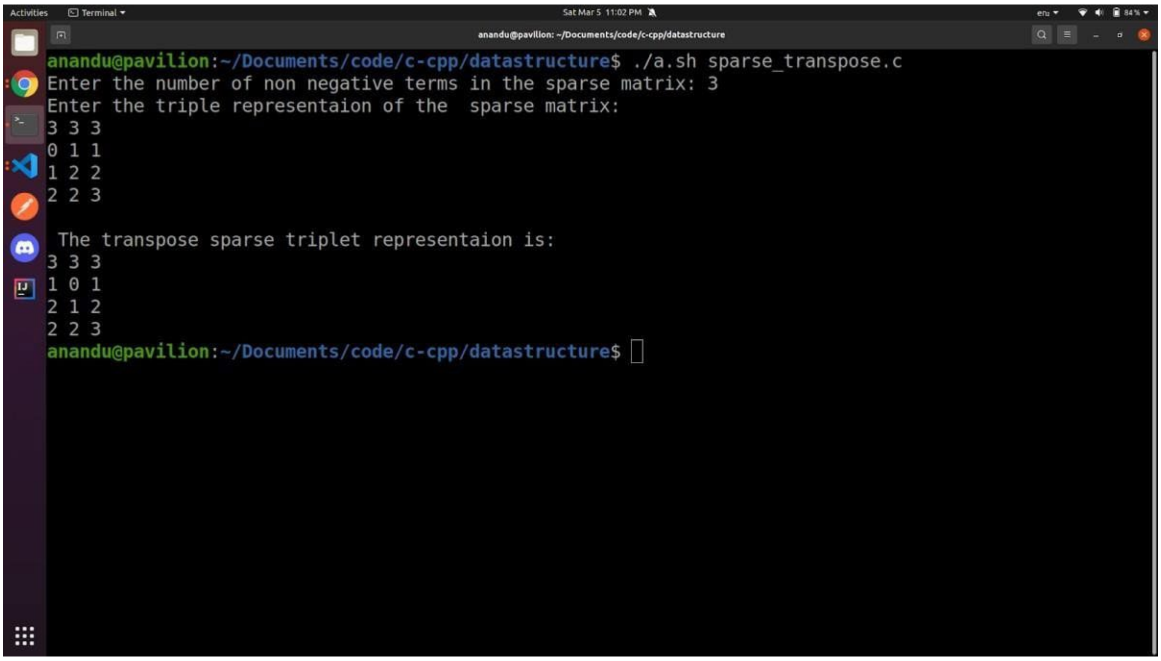
```
                }
            }
            while (i <= n)
            {
                c[k].row = a[i].row;
                c[k].col = a[i].col;
                c[k].val = a[i].val;
                k++;
                i++;
            }
            while (j <= m)
            {
                c[k].row = b[j].row;
                c[k].col = b[j].col;
                c[k].val = b[j].val;

                k++;
                j++;
            }
            c[0].val = k - 1;
        }
        printf("The Sum triple sparse matrix is: \n");
        print(k - 1);
        }
```

```
int main()
{
        printf("Enter the number of non negative terms in the sparse matrix1: ");

        int  n;
        scanf("%d", &n);
        printf("Enter the triple representaion of the sparse matrix1: ");
        int i;
        for (i = 0; i <= n; i++)
        {
                scanf("%d %d %d", &a[i].row, &a[i].col, &a[i].val);
        }
        printf("Enter the number of non negative terms in the sparse matrix2: ");
        int m;
        scanf("%d", &m);
        printf("Enter the triple representaion of the sparse matrix2: ");
        for (i = 0; i <= m; i++)
        {
                scanf("%d %d %d", &b[i].row, &b[i].col, &b[i].val);
        }
        add(n, m);

        return 0;
        }
```

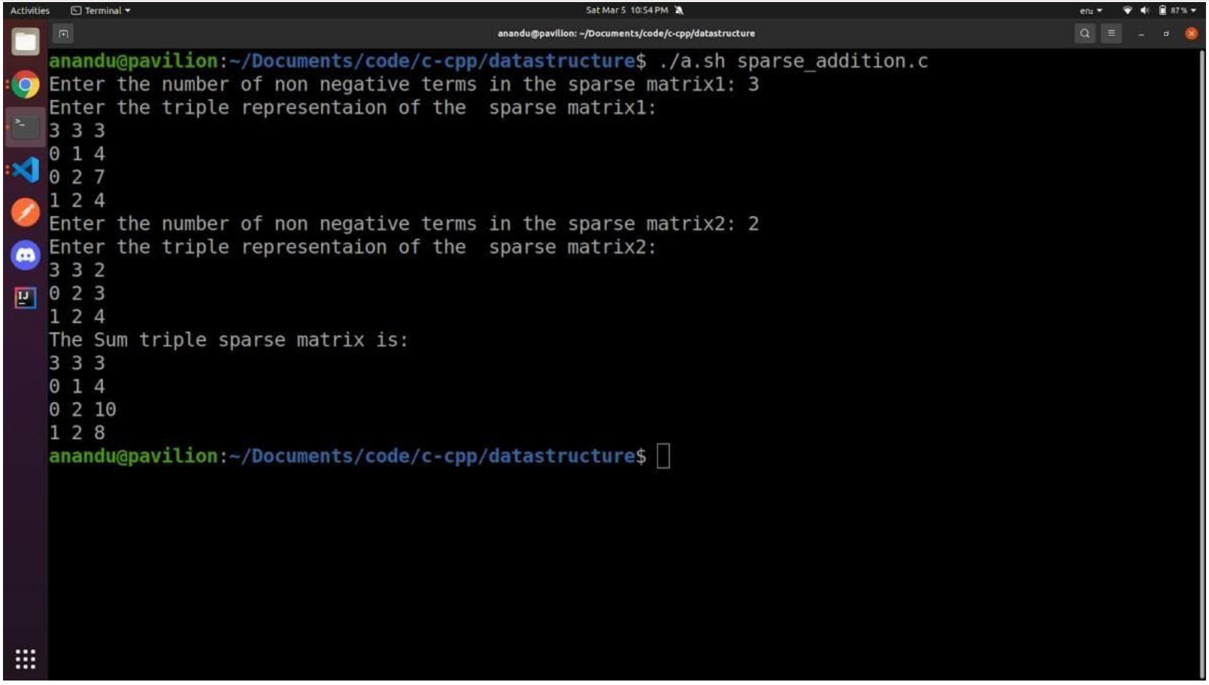
OUTPUT



```
anandu@pavilion:~/Documents/code/c-cpp/datastructure$ ./a.sh triplet_rep_sparse.c
Enter the Order  of the Sparse array: 3 3
Enter the elements: 0 1 0 0 0 1 1 1 0
The Triplet representation is
3 3 4
0 1 1
1 2 1
2 0 1
2 1 1
anandu@pavilion:~/Documents/code/c-cpp/datastructure$ 
```



```
anandu@pavilion:~/Documents/code/c-cpp/datastructure$ ./a.sh sparse_transpose.c
Enter the number of non negative terms in the sparse matrix: 3
Enter the triple representaion of the  sparse matrix:
3 3 3
0 1 1
1 2 2
2 2 3

 The transpose sparse triplet representaion is:
3 3 3
1 0 1
2 1 2
2 2 3
anandu@pavilion:~/Documents/code/c-cpp/datastructure$ 
```



```
anandu@pavilion:~/Documents/code/c-cpp/datastructure$ ./a.sh sparse_addition.c
Enter the number of non negative terms in the sparse matrix1: 3
Enter the triple representaion of the  sparse matrix1:
3 3 3
0 1 4
0 2 7
1 2 4
Enter the number of non negative terms in the sparse matrix2: 2
Enter the triple representaion of the  sparse matrix2:
3 3 2
0 2 3
1 2 4
The Sum triple sparse matrix is:
3 3 3
0 1 4
0 2 10
1 2 8
anandu@pavilion:~/Documents/code/c-cpp/datastructure$ 
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 03

QUEUE

PROBLEM DEFINITION:

Write a program to implement Queue.

ALGORITHM:

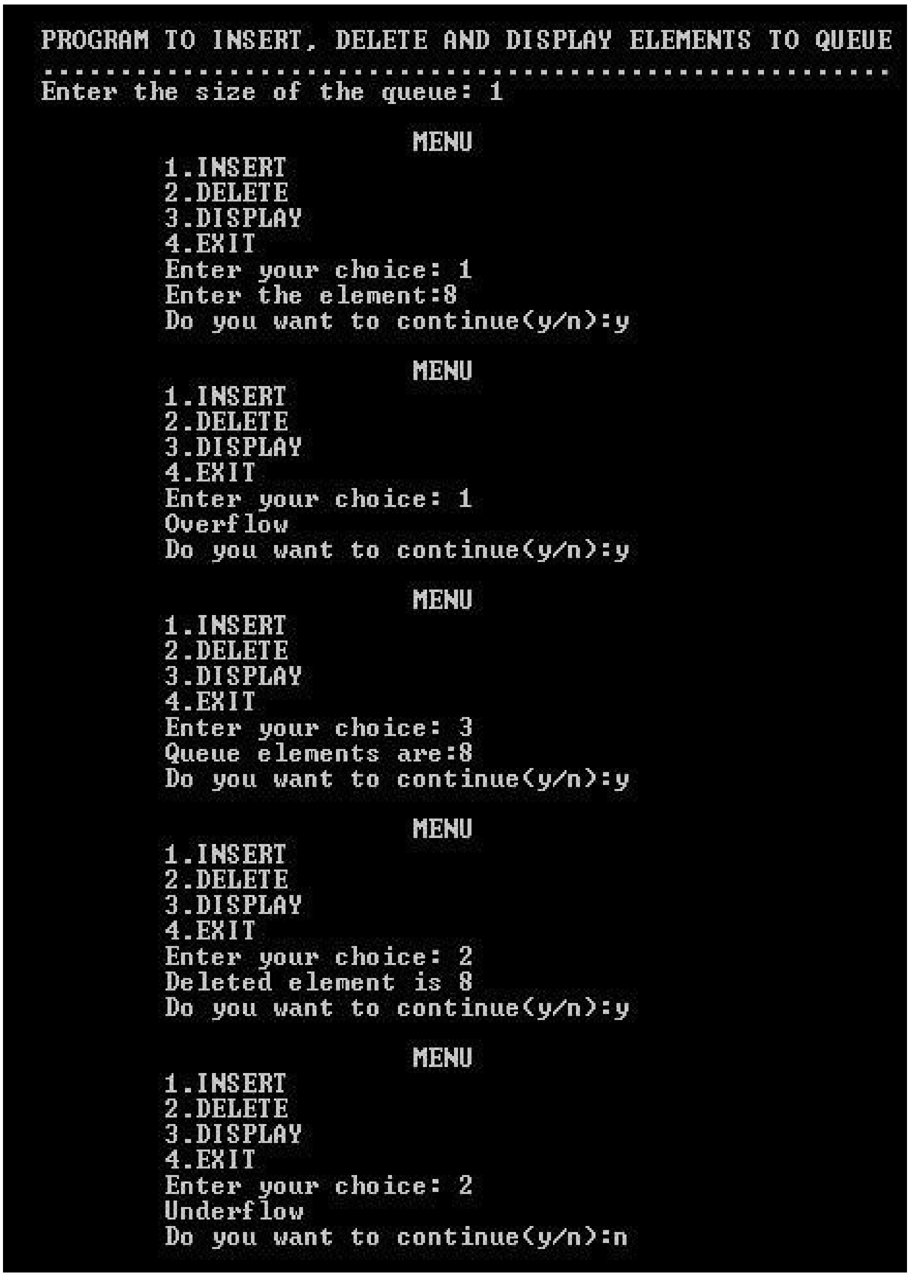
```
Step 1: Start
Step 2: Set front and rear to -1
Step 3: Read size of queue
Step 4: If (insertion)
        Step 4.1: If (rear=size-1)
                Step 4.1.1: Print overflow
        Step 4.2: End if
        Step 4.3: Else
                Step 4.3.1: Print enter the element
                Step 4.3.2: Read the element
                Step 4.3.3: If (front=-1& rear=-1)
                        Step 4.3.3.1: Set front=0
                        Step 4.3.3.2: Increment rear
                        Step 4.3.3.3: Read element to queue [rear]
                        Step 4.3.3.4: Endif
                Step 4.3.4: End else
Step 5: If (deletion)
        Step 5.1: If (front=-1)
                Step 5.1.1: Print underflow
        Step 5.2: End if
        Step 5.3: Else
                Step 5.3.1: Print deleted element is queue [front]
                Step 5.3.2: If (front=rear)
                        Step 5.3.2.1: Set front and rear as -1
                        Step 5.3.2.2: Endif
                        Step 5.3.2.3: Else
                                Step 5.3.2.3.1: Increment front
                                Step 5.3.2.3.2: End Else
        Step 5.4: Endif
Step 6: If (display)
        Step 6.1: If (front & rear are -1)
                Step 6.1.1: Print underflow
                Step 6.1.2: End if
Step 6.1.3: Else
        Step 6.1.3.1: Set i as zero Step 6.1.3.2: Loop
        (i<rear)
                Step 6.1.3.2.1: Print queue[i] Step 6.1.3.2.2: Loop
                ends
Step 6.1.4: End else

Step 7: End if

Step 8: Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int no,i,front=-1,rear=- 1,k,queue[50],element,n; char c; clrscr();
printf("\n\n\n\tPROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO QUEUE");
printf("\n\t  \n\n
\n"); printf("\n\n\tEnter the size of the queue: "); scanf("%d",&n
); do
{
printf("\n\n\n\t\t\t\tMENU\n\n");
printf("\t\t1.INSERT\n\n\t\t2.DELETE\n\n\t\t3.DISPLAY\n\n\t\t4.EXIT\n\n\t\tEnter your choice:
"); scanf("%d",&no);
if(no==1)
{
if(rear==(n-1))
printf("\n\t\tOverfl
ow"); else {
printf("\n\n\t\tEnter the element: ");
scanf("%d",&element); if(front==- 1&&rear==-1)
front=0;
rear++;
queue[rear]=element;
}
}
if(no==2)

{ if(front==-1)
printf("\n\t\tUnderflow
\n");
else
{
```

```
k=queue[front];
printf("\n\t\tDeleted element is %d ",k);
}
if(front== rear)
front=rear
=-1; else
front++;
}
if(no==3)
{
if(front==-1&&rear==-1)
printf("\n\t\tUnderflow\n");
else
{
printf("\n\t\tQueue elements are\n ");
for(i=front;i<=rear;i++)
{
printf("\n\n\t\t%d",queue[i]);
}
}
}
if(no==4)
break;
printf("\n\n\t\tDo you want to continue(y/n) ");
scanf(" %c",&c);
}
while(c=='y'||c==' Y'); getch();
}
```

OUTPUT:



```
PROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO QUEUE
.......................................................
Enter the size of the queue: 1

                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 1
        Enter the element:8
        Do you want to continue(y/n):y

                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 1
        Overflow
        Do you want to continue(y/n):y

                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 3
        Queue elements are:8
        Do you want to continue(y/n):y

                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 2
        Deleted element is 8
        Do you want to continue(y/n):y

                        MENU
        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 2
        Underflow
        Do you want to continue(y/n):n
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 04

STACK

PROBLEM DEFINITION:

Write a program to implement Stack.

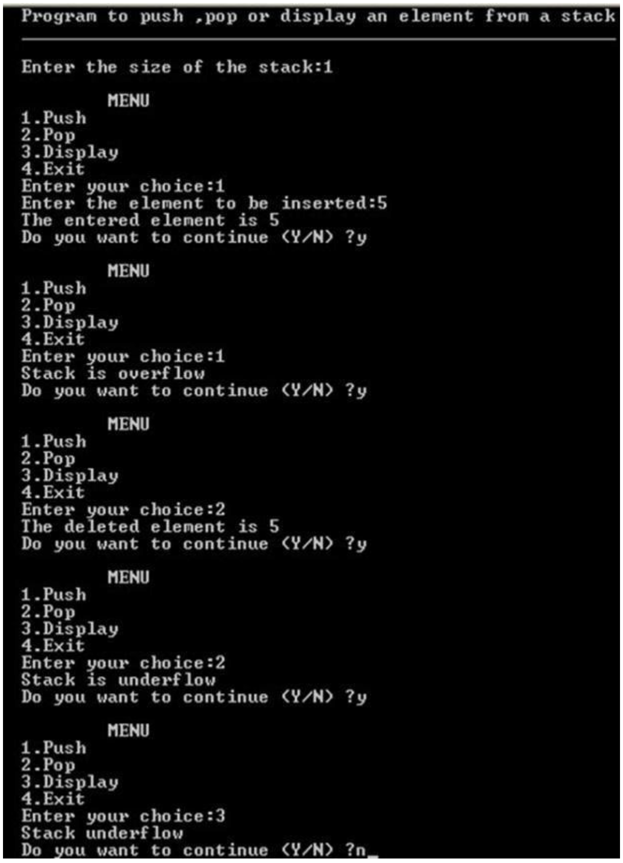
ALGORITHM:

```
Step 1: Start
Step 2: Set top to -1
Step 3: Print size of stack
Step 4: If (push)
        Step 4.1: If (top=size-1)
                Step 4.1.1: Print overflow
                Step 4.1.2: Endif
                Step 4.1.3: Else
                        Step 4.1.3.1: Decrement top
                        Step 4.1.3.2: Read one element
                        Step 4.1.3.3: Set stack[top] to element
                Step 4.1.4: Else end
Step 5: If (pop)
        Step 5.1: If (top=-1)
                Step 5.1.1: Print overflow
                Step 5.1.2: Endif
                Step 5.1.3: Else
                        Step 5.1.3.1: Set item to stack[top]
                        Step 5.1.3.2: Set top to top-1
                        Step 5.1.3.3: Else end
Step 6: If (display)
        Step 6.1: Set i as top
        Step 6.2: Loop (i<top)
                Step 6.2.1: Print stack
                Step 6.2.2: Increment i
 Step 6.3: Loop ends Step
7: Stop
```

PROGRAM DEVELOPMENT:

```
        #include<stdio.h
        #include<conio.h>
void main()
        {
        int st[50],top=-1,k,n,i,si;
        char ch;
        clrscr();
        printf("\n\tProgram to push ,pop or display an element from a stack");
        printf("\n\t__________________________________________________________________\n\
        n"); printf("\n\tEnter the size of the stack:") ;
        scanf("%d",&si
        ); do
        {
        printf("\n\t\tMENU");
        printf("\n\t\t_____\n");
        printf("\n\n\t1.Push\n\n\t2.Pop\n\n\t3.Display\n\n\t4.Exit");
        printf("\n\n\tEnter your choice:");
        scanf("%d",&n);
        if(n==1)
        {
        if(top==si-1)
        {
        printf("\n\tStack is
        overflow"); else {
        printf("\n\tEnter the element to be inserted:");
        scanf("%d",&
        k); top++;
        st[top]=k;
        printf("\n\tThe entered element is %d",st[top]);
        }
        }
        else if(n==2)
        {
        if(top==-1)
        {
        printf("\n\tStack is underflow");
        }
        else
        {
        printf("\n\tThe deleted element is %d",st[top]);
        top--;
        }
        }

        else if(n==3)

        {
        if(top==-1)
        {
        printf("\n\tStack underflow");
        } else {
        printf("\n\tThe stack
        is:");
        for(i=top;i>=0;i--)
        {
        printf("\t\t%d",st[i]);
        }
        }}
        else

        {
        break;

        }
        printf("\n\n\tDo you want to continue (Y/N) ?");
        scanf(" %c",&ch);
        }
        while((ch=='Y')||(ch=='y'));
        getch();
        }
```

OUTPUT:



```
Program to push ,pop or display an element from a stack
-------------------------------------------------------

Enter the size of the stack:1

        MENU
1.Push
2.Pop
3.Display
4.Exit
Enter your choice:1
Enter the element to be inserted:5
The entered element is 5
Do you want to continue (Y/N) ?y

        MENU
1.Push
2.Pop
3.Display
4.Exit
Enter your choice:1
Stack is overflow
Do you want to continue (Y/N) ?y

        MENU
1.Push
2.Pop
3.Display
4.Exit
Enter your choice:2
The deleted element is 5
Do you want to continue (Y/N) ?y

        MENU
1.Push
2.Pop
3.Display
4.Exit
Enter your choice:2
Stack is underflow
Do you want to continue (Y/N) ?y

        MENU
1.Push
2.Pop
3.Display
4.Exit
Enter your choice:3
Stack underflow
Do you want to continue (Y/N) ?n_
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 05

CIRCULAR QUEUE

PROBLEM DEFINITION:

Write a program to implement Circular Queue.

ALGORITHM:

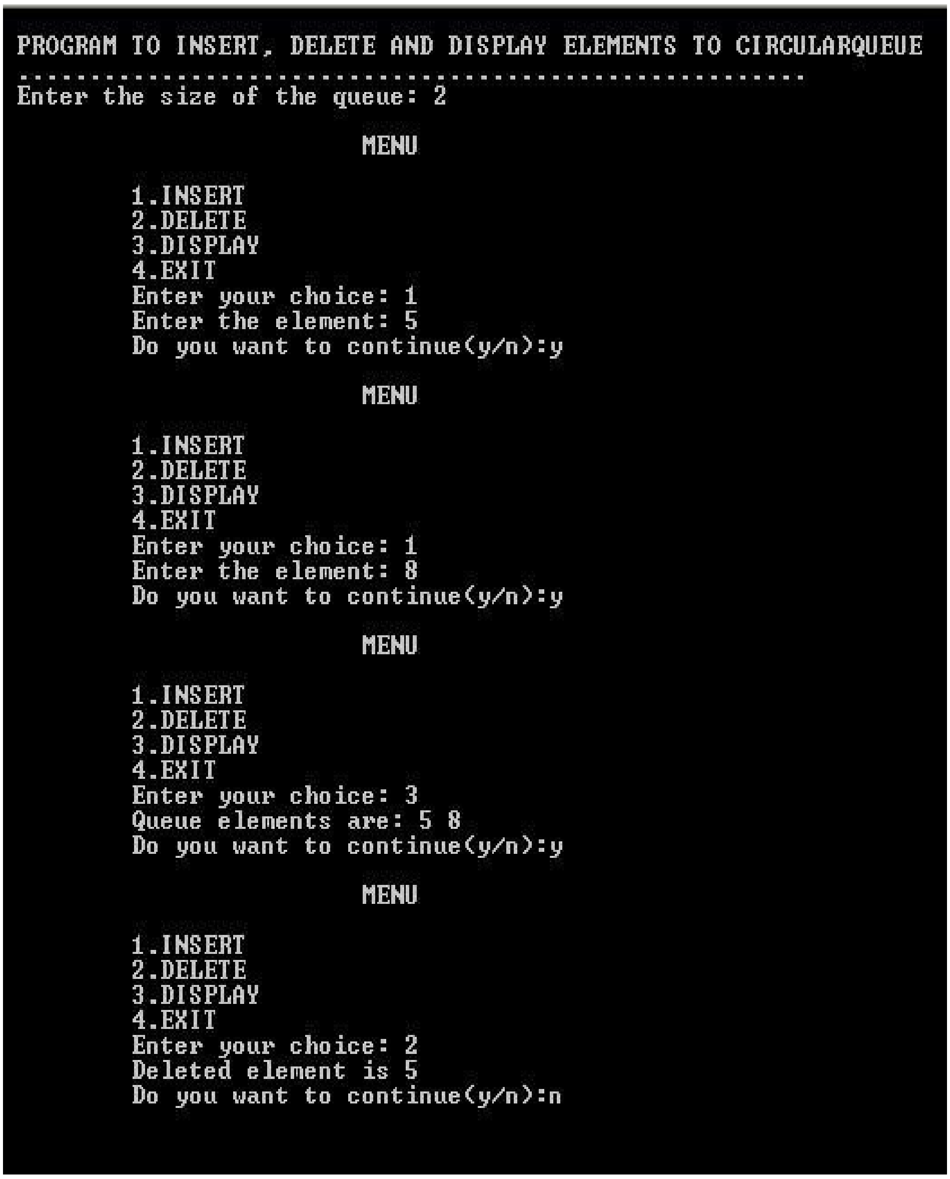
```
Step 1: Start
Step 2: Set front=0&rear=0
Step 3: Read size of queue
Step 4: If (insertion)
        Step 4.1: Read the element
        Step 4.2: If (front and rear are zero)
                Step 4.2.1: Set front&rear to 1
                Step 4.2.2: Read element to queue [rear]
        Step 4.3: Endif
        Step 4.4: Else
                Step 4.4.1: Set next as rear mod size+1
                Step 4.4.2: If (next=front)
                        Step 4.4.2.1: Print overflow
                        Step 4.4.2.2: End if
                Step 4.4.3: Else
                        Step 4.4.3.1: Set rear as next
                        Step 4.4.3.2: Read element to queue [rear]
                        Step 4.4.3.3: End else
                Step 4.4.5: End else
Step 5: End if
Step 6: If (deletion)
        Step 6.1: If (front=0)
        Step 6.2: Print underflow
        Step 6.3: Endif
        Step 6.4: Else
                Step 6.4.1: Print deleted element as queue [front]
                Step 6.4.2: If (front =rear)
                Step 6.4.3: Set front and rear as 0
                Step 6.4.4: Endif
        Step 6.5: Else
                Step 6.5.1: Set front as front mmode size+1
                Step 6.5.2: End else
                Step 6.5.3: End else
Step 7: End  if
Step 8: If (display)
    Step 8.1: If (front & rear =0)
        Step 8.1.1: (Print under flow)
    Step 8.2: End if
    Step 8.3: Else
        Step 8.3.1: If (front <rear)
            Step 8.3.1.1: Set i as front
            Step 8.3.1.2: Loop (i<rear)
                    Step 8.3.1.2.1: Print queue[i]
                    Step 8.3.1.2.2: Increment i
            Step 8.3.1.3: Loop ends
        Step 8.3.2: Endif
        Step 8.3.3: Else
            Step 8.3.3.1: Loop (i<rear)
                Step 8.3.3.1.1: Print queue[i]
                Step 8.3.3.1.2: Increment i
            Step 8.3.3.2: Loop ends
            Step 8.3.3.3: Set I as 1
            Step 8.3.3.4: Loop (i<rear)
                Step 8.3.3.4.1: Print queue[i]
                Step 8.3.3.4.2: Increment i
            Step 8.3.3.5: Loop ends
        Step 8.3.4: Else end
Step 9: Endif
Stop 10: Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int
no,i,front=0,rear=0,k,queue[50],element,size,next=
1; char c; clrscr();
printf("\n\tPROGRAM     TO     INSERT,       DELETE       AND    DISPLAY
    ELEMENTS    TO CIRCULARQUEUE");
printf("\n\t                                        ");
printf("\n\tEnter the size of the queue: ");
scanf("%d",&size
);
 do
```

```
{
printf("\n\t\t\t\tMENU\n\n");
printf("\t\t1.INSERT\n\t\t2.DELETE\n\t\t3.DISPLAY\n\t\t4.EXIT\n\t\tEnter your choice:
"); scanf("%d",&no);
if(no==1)
{
if(rear==size-1)
{
printf("\n overflow ");
}
printf("\t\tEnter the element: ");
scanf("%d",&element);
if(front==0&&rear==0)
{
front=rear=1;
queue[rear]=element;
}
else
{
next=(rear%size)+1;
if(next==front)
printf("\t\tOverflow\n");
else
{
rear=next;
queue[rear]=element;
}
}
}
if(no==2)
{ if(front==0)
printf("\t\tUnderflow
\n");
else
{
k=queue[front];
printf("\t\tDeleted element is %d\n",k);
}
if(front==re
ar)
front=rear=0;
else
front=(front%size)+1;
if(no==3)
{
if(front==0&&rear==0)
printf("\t\tUnderflow\n");
else
{
printf("\t\tQueue elements are");
if(front<rear)
{
for(i=front;i<=rear;i++)
printf("\n\t%d\t",queue[i]);
}
else
{
for(i=front;i<=size;i++)
printf("\n\t\t%d",queue[i]);
for(i=1;i<=rear;i++)
printf("\n\t\t%d",queue[i]);
}
}
}
if(no==4)
break;
printf("\t\tDo you want to continue(y/n):");
scanf(" %c",&c);
}
while(c=='y'||c==’ Y');
 getch();
}
```

OUTPUT:



```
PROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO CIRCULARQUEUE
.......................................................
Enter the size of the queue: 2

                        MENU

        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 1
        Enter the element: 5
        Do you want to continue(y/n):y

                        MENU

        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 1
        Enter the element: 8
        Do you want to continue(y/n):y

                        MENU

        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 3
        Queue elements are: 5 8
        Do you want to continue(y/n):y

                        MENU

        1.INSERT
        2.DELETE
        3.DISPLAY
        4.EXIT
        Enter your choice: 2
        Deleted element is 5
        Do you want to continue(y/n):n
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 06

PRIORITY QUEUE

PROBLEM DEFINITION:

Write a program to implement priority queue.

ALGORITHM:

```
Step1: Start
Step 2: To enqueue:
        If front=0 && rear=size-1
                Print Queue full
        Else if Front=-1
                Front=0, Rear=0
                A[Rear].item=ITEM
                A[Rear].prio=PRORITY
        Else if Rear=size-1
                For i=Front to Rear
                        A[i-1]=A[i]
                Front=Front-1
                Rear=Rear-1
                For i=Rear to Front
                        If(A[i].prio<PRIORITY)
                                break;
                        loc=i+1
                for i=Rear to loc
                        A[i+1]=A[i]
                A[loc].item=ITEM
                A[loc].prio=PRIORITY
                Rear=Rear+1
        Else
                For i=Rear to Front
                        If(A[i].prio<PRIORITY
                                break
                        loc=i+1
                        for i=Rear to loc
```

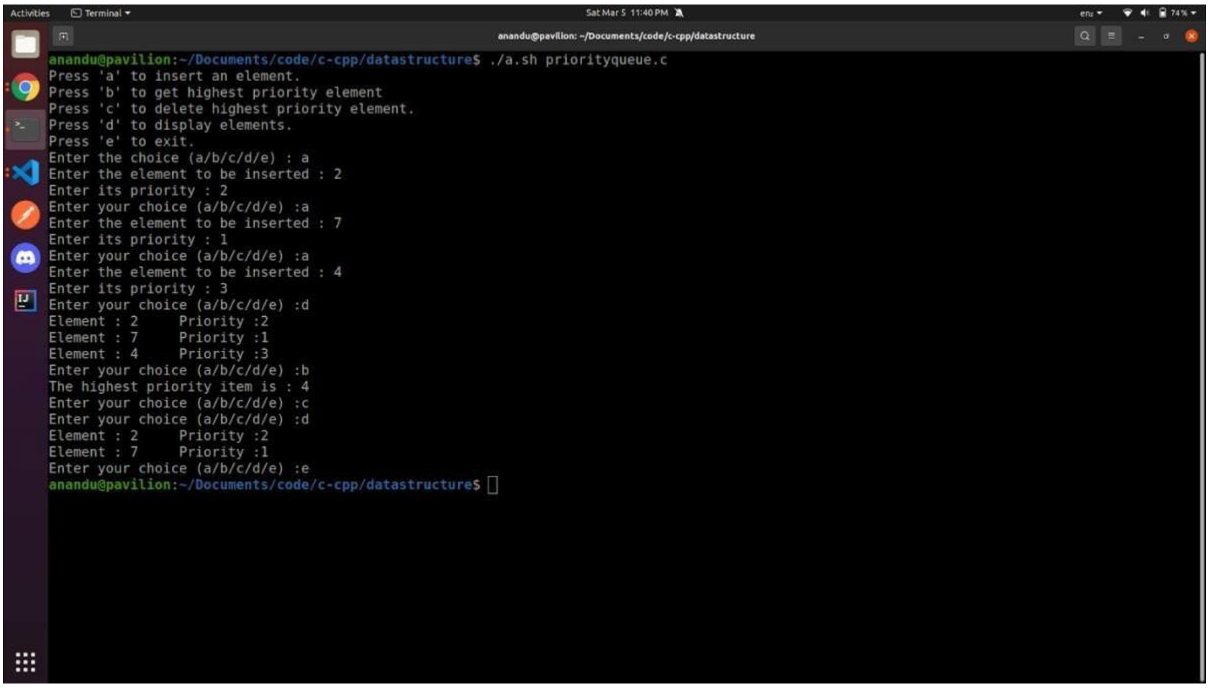
```
                    A[i+1]=A[i]
                    A[loc].item=ITEM
                    A[loc].prio=PRIORITY
                    Rear=Rear+1
Step 3: To delete
          If Front =-1
                    Print Queue empty
          Else if Rear=Front
                    Front=-1, Rear=-1
          Else      Step 4:Stop
                                                       Front++
```

PROGRAM DEVELOPMENT:

```
#include <stdio.h>
#define SIZE 5
typedef struct
{
          int element;
          int priority;
} priorq;

priorq pq[SIZE];
int F = -1, R = -1;
void insertpq(int, int);
void display(void);
int  get_highest_priority(void);
void delete_highest_priority(void);
void insertpq(int item, int prior)
{
          if (R >= SIZE - 1)


int main()
{
```

```
            char ch;
            int item, prior, max;
            printf("Press 'a' to insert an element.\n");

{ printf("Queue full!\n");
            }
            else

    { if (F == -1 && R == -1)
    {
          F=0;
          R++;
                        }


                        pq[R].element = item;
                        pq[R].priority = prior;
            }
 }

 int get_highest_priority(void)
 {
            int maxp = 0, maxi, i;
            for (i = F; i <= R; i++)
            {
                        if (pq[i].priority > maxp)
                        {
                                    maxp = pq[i].priority;
                                    maxi = pq[i].element;
                        }
            }
            return maxi;}

 void delete_highest_priority(void)
 {
            int m, i, j;
            m = get_highest_priority();
            if (R == -1)
            {
                    printf("Queue empty\n");
            }
            else
            {
    for (i = F; pq[i].element != m; i++)
    {
    for (j = i; j < R; j++)
    {
1].element; 1].priority;}


  }


  void display(void)
  {
            int i;
            if (R == -1)
            {
                    printf("Queue empty\n");
            }
            else
            {
    for (i = F; i <= R; i++)
                            { printf("Element : %d\tPriority
                            :%d\n", pq[i].element,
                            pq[i].priority);
                  }
            } }
            printf("Press 'b' to get highest priority
  element\n");
            printf("Press 'c' to delete highest priority
  element.\n");
            printf("Press 'd' to display elements.\n");
            printf("Press 'e' to exit.\n");
            printf("Enter the choice (a/b/c/d/e) : ");
            do

  { scanf("%c", &ch); switch (ch)
  {
  case 'a':
            printf("Enter the element to be inserted”);
            scanf("%d", &item); printf("Enter its priority :");
            scanf("%d", &prior); insertpq(item, prior);
            break;
case 'b':
            max = get_highest_priority();
            printf("The highest priority
item is : %d\n", max);
break;


case 'c':
          delete_highest_priority(); break;
case 'd':
          display(); break;
case 'e':
          break; default:
  printf("Enter your choice(a/b/c/d/e) :");
```

OUTPUT:



```
anandu@pavilion:~/Documents/code/c-cpp/datastructure$ ./a.sh priorityqueue.c
Press 'a' to insert an element.
Press 'b' to get highest priority element
Press 'c' to delete highest priority element.
Press 'd' to display elements.
Press 'e' to exit.
Enter the choice (a/b/c/d/e) : a
Enter the element to be inserted : 2
Enter its priority : 2
Enter your choice (a/b/c/d/e) :a
Enter the element to be inserted : 7
Enter its priority : 1
Enter your choice (a/b/c/d/e) :a
Enter the element to be inserted : 4
Enter its priority : 3
Enter your choice (a/b/c/d/e) :d
Element : 2     Priority :2
Element : 7     Priority :1
Element : 4     Priority :3
Enter your choice (a/b/c/d/e) :b
The highest priority item is : 4
Enter your choice (a/b/c/d/e) :c
Enter your choice (a/b/c/d/e) :d
Element : 2     Priority :2
Element : 7     Priority :1
Enter your choice (a/b/c/d/e) :e
anandu@pavilion:~/Documents/code/c-cpp/datastructure$ 
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 07

DEQUEUE

PROBLEM DEFINITION:

Write a program to implement Dequeue.

ALGORITHM:

```
Step 1: Start
Step 2: Read choice
Step 3: If (insert front)
        Step 3.1: If front =0
                Step 3.1.1: Print overflow
        Step 3.2: Else
                Step 3.2.1: Read element
        Step 3.3: End if
        Step 3.4: If (front= -1)
                Step 3.4.1: Set front =rear=0
                Step 3.4.2: Insert element
        Step 3.5: Else if (front!=0)
                Step 3 5.1: Set front= front-1
                Step 3.5.2: Insert element
        Step 3.6: End if
Step 4: If (insert end)
        Step 4.1: If (rear=
        n-1)
              Step 4.1.1: Print 'overflow'
        Step 4.2: Else
                Step 4.2.1: Read element
                Step 4.2.2: If (front =-1)
                        Step 4.2.2.1: Set front=rear=0
                Step 4.2.3: Else
                        Step 4.2.3.1: Rear+ +
                Step 4.2.4: End if
                Step 4.2.5: Insert element
        Step 4.3: End if
Step 5: If (delete front) Step 5.1: If
        (f= -1) Step 5.1.1: Print
        overflow
        Step 5.2: Else
                Step 5.2.1: If (f=rear)
```

```
                    Step 5.2.2: F=rear=-1
          Step 5.3: End if
Step 6: If (insertion at end)
          Step 6.1: If (f=-1)
                    Step 6.1.1: Print under flow
          Step 6.2: Else
                    Step 6.2.1: If (f=rear)
                              Step 6.2.1.1: F=rear=-1
                    Step 6.2.2: End if
                    Step 6.2.3: Rrear =rear-1
          Step 6.3: End if
Step 7: Stop.
```

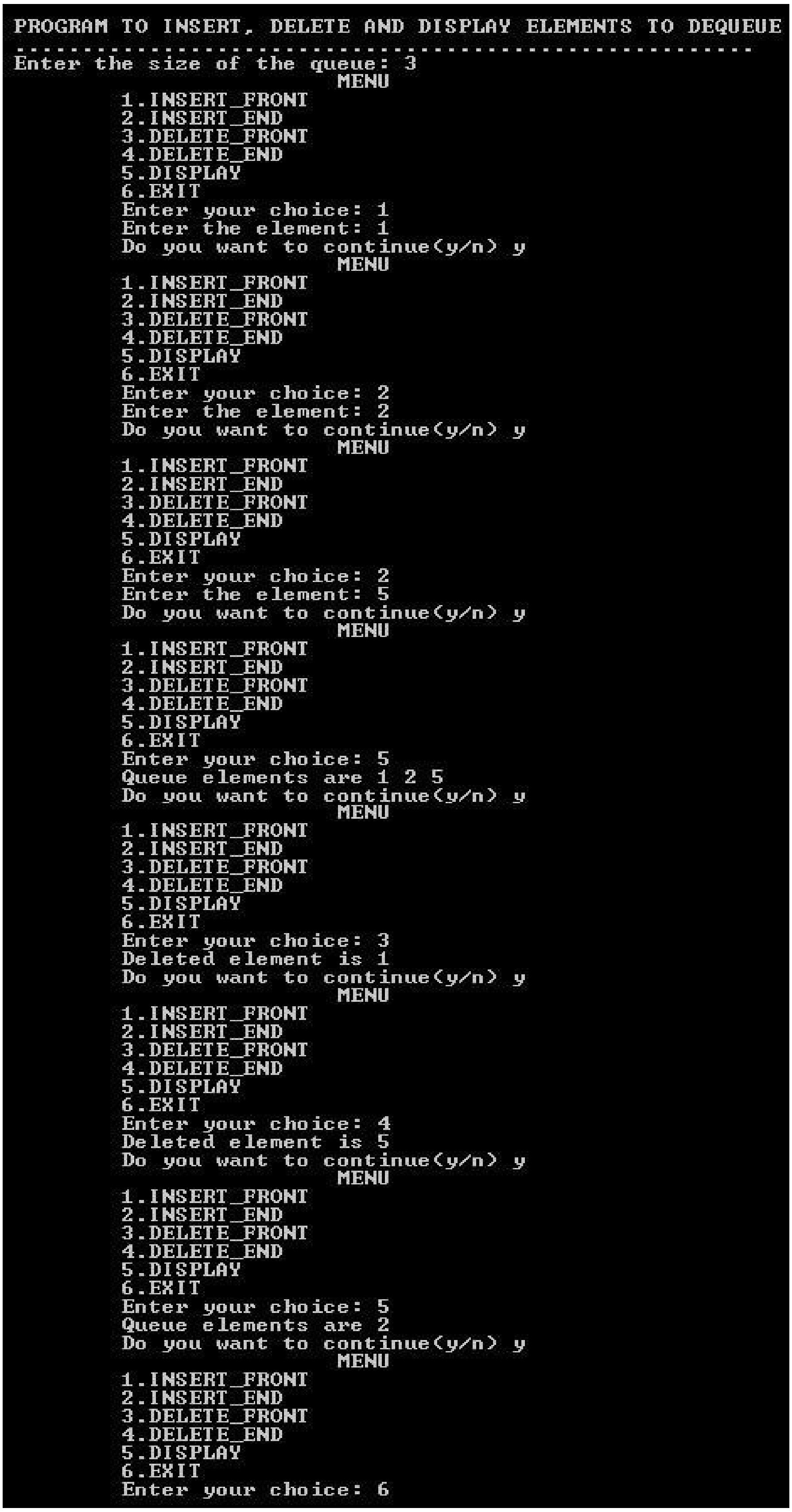
PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
void main()
{
Int no,i,front=-1,rear=1,k,queue[50],element,n; char c; clrscr();
printf("\n\tPROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO
DEQUEUE\n");
printf("\t ");
printf("\n\tEnter the size of the queue: ");
scanf("%d",&n
); do
{
printf("\t\t\t\tMENU\n");
printf("\t\t1.INSERT_FRONT\n\t\t2.INSERT_END\n\t\t3.DELETE_FRONT\n\t\t4.DELET
E_E
ND\n\t\t5.DISPLAY\n\t\t6.EXIT\n\t\tEnter your choice: ");
scanf("%d",&no);
if(no==1)
{ if(front==0)
printf("\t\tOverflow
\n"); else {
printf("\t\tEnter the element: ");
scanf("%d",&element);
if(front==-1)
{front=0;
rear=0;
queue[front]
=element;
```

```
    } else
    if(front!=0)
    { front=front-1;
    queue[front]=ele
    ment;
    }
    }
    }
    if(no==2)
    {
    if(rear==n-1) printf("\t\tOverflow\n");
    else { printf("\t\tEnter the element: ");
    scanf("%d",&element); if(front==-1)
    {
    front=rear=0;
    queue[rear]=element;
}

   else

    {
    rear=rear+1;
    queue[rear]=element;
    }
    }
    }
    if(no==3)
    { if(front==-1) printf("\t\tUnderflow
    \n"); else
    {
    k=queue[front];
    printf("\t\tDeleted element is %d
    ",k); printf("\n");
    if(front==rear) front=rear=-1;
    else
    front++;
    }
    }
    if(no==4)
    {
    if(rear==-1)
    printf("\t\tUnderflow\n");
```

```
else
{
k=queue[rear];
printf("\t\tDeleted element is %d
",k);
 printf("\n");
if(front==rear)
front=rear=-1;
else
rear--; }
}
if(no==5)
{
if(front==-1&&rear==-1)
printf("\t\tUnderflow\n");
else {

printf("\t\tQueue elements are
");
 for(i=front;i<=rear;i++)
printf("%d ",queue[i]);
printf("\n");
}
}
if(no==6)
break;
printf("\t\tDo you want to continue(y/n) ");
scanf(" %c",&c);
}
while(c=='y'||c=='Y');
 getch();
}
```

OUTPUT:



```
PROGRAM TO INSERT, DELETE AND DISPLAY ELEMENTS TO DEQUEUE
........................................................
Enter the size of the queue: 3
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 1
        Enter the element: 1
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 2
        Enter the element: 2
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 2
        Enter the element: 5
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 5
        Queue elements are 1 2 5
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 3
        Deleted element is 1
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 4
        Deleted element is 5
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 5
        Queue elements are 2
        Do you want to continue(y/n) y
                        MENU
        1.INSERT_FRONT
        2.INSERT_END
        3.DELETE_FRONT
        4.DELETE_END
        5.DISPLAY
        6.EXIT
        Enter your choice: 6
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 08

INFIX TO POSTFIX EVALUATION

PROBLEM DEFINITION:

Write a program to implement Infix to Postfix Evaluation.

ALGORITHM:

```
Step 1: Start
Step 2: Read the expression
Step 3: For I from 0 to length
    Step 3.1: Set ch as a[i]
    Step 3.2: If (ch=’c’)
        Step 3.2.1: Push (ch)
    Step 3.3: If (ch=’)’)
        Step 3.3.1: While (stk [top]!=’(’)
            Step 3.3.1.1: Read stk [top]=post[j++]
            Step 3.3.1.2: Decrement top
        Step 3.3.2: End loop
        Step 3.3.3: Decrement top
    Step 3.4: If (ch=’+’ or ch=’-’ or ch=’*’ orch=’/’)
        Step 3.4.1: If (top=-1 or stk[top]=’(’)
            Step 3.4.1.1: Push (ch)
        Step 3.4.2: Else
            Step 3.4.2.1: x=priority (ch)
            Step 3.4.2.2: y=priority (stk [top])
            Step 3.4.2.3: If(y>=x)
                Step 3.4.2.3.1: Read stk [top] to post [j++]
                Step 3.4.2.3.2: Decrement top
                Step 3.4.2.3.3: Push (ch)
            Step 3.4.2.4: Else
                Step 3.4.2.4.1: Push (ch)
            Step 3.4.2.5: End if
        Step 3.4.3:  End  if
    Step 3.5: If (ch is an alphabet)
        Step 3.5.1: Read ch to post [j++]
    Step 3.6: End if
Step 4: End loop
Step 5: While (stk [top] !=’\0’)
    Step 5.1: Read stk [top] to post [j++]
```

```
        Step 5.2: Decrement top

Step 6: End loop
Step7: Assign post[i] as ‘\0’
Step 8: Print “Post fix expression as post”
Step9: Stop
```

Eval-Postfix ()

```
Step 1: Start
Step 2: Find postfix expression for given expression
Step 3: For I from 0 to length of post
        Step 3.1: Set ch as post[i]
        Step 3.2: If ch is an alphabet
           Step 3.2.1: Print “Enter the value for ch”
    Step 3.2.2: Read the value to c.
  Step 3.3.3: Push C to another stk. Step 3.3:
Else
                Step 3.3.1: Set o1 as stk [top]
                Step 3.3.2: Decrement top
                Step 3.3.3: Set o2 as stk [top]
                Step 3.3.4: Decrement top
                Step 3.3.5: If(ch== +)
                        Step 3.3.5.1: x=o1+o2
                Step 3.3.6: If(ch= -)
                        Step 3.3.6.1: x=o1-o2
                Step 3.3.7: If (ch=*)
                        Step 3.3.7.1: x=o1*o2
                Step 3.3.8: If (ch=/)
                        Step 3.3.8.1: x=o1/o2
                Step 3.3.9: End if
                Step 3.3.10: Push (x)
        Step 3.4: End if
Step 4: End for
Step 5: Print value as stk [top]
Step 6: Stop
```

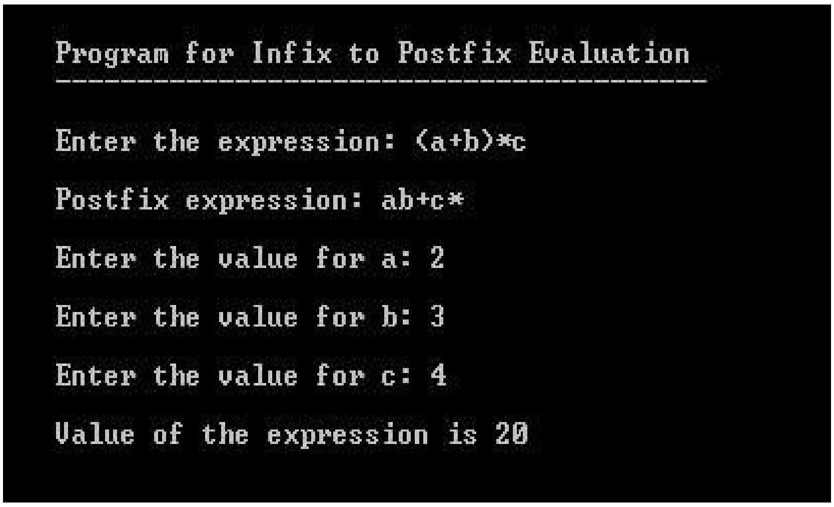
PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void push(char);
```

```
void
push1(int); int
priority(char);
void read();
int top=-1,top1=-1,j=0,i,x,y; char
stk[50],stk1[50],a[50],ch,post[50];
void main()
{
clrsc
r();
printf("\n\n\tProgram for Infix to Postfix Evaluation"); printf("\n\t------------------------------
----------\n"); printf("\n\tEnter the expression: ");
gets(a);
for(i=0;a[i]!='\0';i++)
{
ch=a[i];
switch(ch) {
case
'(':push(ch);
break;
case')':while (stk[top]!='(')
{
post[j++]=stk[ top]; top--; } top--; break;
case '+': case '-
': case '^': case '/':
case '*': if (top== -1||stk[top]=='(')
push(ch); else
{x=priority(ch);
y=priority(stk[t
op]); if(y>=x)
{
post[j++]=stk[
top]; top--;
push(ch);} else
push(ch);
}
break;
default:
if(isalpha(ch))
 post[j++]=ch;
break;
}
}
while(stk[top]!='\0')
{
post[j++]=stk[top];
top--; }
post[j]
='\0';

printf("\n\tPostfix
expression: ");
puts(post);
read();
 getch();
}
void push(char ch)
{
top++;
stk[top]=ch;
} void
push1(int ch)
{top1++;
stk1[top1]=c
h;
}
int priority(char c) {
if (c=='t'||c=='-')
return 1;
elseif(c=='*'||c=='/')
return 2;
else if(c=='^')
 return 3;
 else
 return 0;
}
 void read ()
{
int c,o1,o2;
for(i=0;post[i]!='\0';i+
+)
{
ch=post[i];
if(isalpha(ch))
```

```
{printf("\n\tEnter the value for %c: ",ch);
scanf("%d", &c);
push1(c);
}
else {
o1=stk1[top1];
top1--;
o2=stk1[top1];
top1--;
switch(ch)
{
case '+':
x=o1+o2;
 break;
 case'-':
x=o1-o2;
 break;
 case'*':
x=o1*o2;
 break;
 case'/':
x=o1/o2;
break;
case'^':
x=o1^o2;
break;
default:
 break;
}
push1
(x);
}
}
printf("\n\tValue of the expression is
%d",stk1[top1]); }

}
```

OUTPUT:



```
Program for Infix to Postfix Evaluation
----------------------------------------

Enter the expression: (a+b)*c

Postfix expression: ab+c*

Enter the value for a: 2

Enter the value for b: 3

Enter the value for c: 4

Value of the expression is 20
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 09

INFIX TO PREFIX CONVERSION

PROBLEM DEFINITION:

Write a program to implement Infix to Prefix Conversion.

ALGORITHM:

Step 1: Start

Step 2: Reverse the infix string. Note that while reversing the string you must interchange left and right parentheses.

Step 3: Obtain the postfix expression of the infix expression Step 1.
Step 4: Reverse the postfix expression to get the prefix expression
Step 5: Stop

PROGRAM DEVELOPMENT:

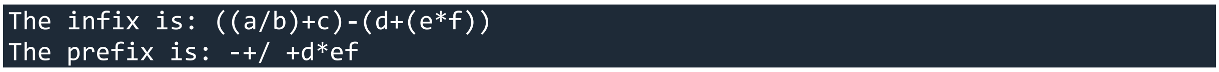
```
#include<stdio.h>

#include<string.h>

#include<limits.h>

#include<stdlib.h>

# define MAX 100

int top = -1;

char stack[MAX];


// checking if stack is full

int isFull() {

return top == MAX - 1;
}


// checking is stack is empty

int isEmpty() {

    return top == -1;

}


// Push function here, inserts value in stack and increments stack top by 1

void push(char item) {

    if (isFull())

      return;

      top++;

      stack[top] = item;

}


// Function to remove an item from stack. It decreases top by 1

int pop() {

    if  (isEmpty())

        return INT_MIN;


    // decrements top and returns what has been popped

    return stack[top--];

}


// Function to return the top from stack without removing it
```

```
int peek(){
    if  (isEmpty())
        return INT_MIN;
    return stack[top];
}

// A utility function to check if the given character is operand
int checkIfOperand(char ch) {
    return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z');
}

// Fucntion to compare precedence
// If we return larger value means higher precedence
int precedence(char ch)
{
    switch (ch)
    {
    case '+':
    case '-':
        return 1;

    case '*':
    case '/':
        return 2;
    case '^':
        return 3;
    }
    return -1;
}

// The driver function for infix to postfix conversion
int getPostfix(char* expression)
{
    int i, j;

    for (i = 0, j = -1; expression[i]; ++i)
    {
        // Here we are checking is the character we scanned is operand or not
        // and this adding to to output.
        if (checkIfOperand(expression[i]))
            expression[++j] = expression[i];

        // Here, if we scan character ‘(‘, we need push it to the
        stack. else if (expression[i] == '(')
            push(expression[i]);

        // Here, if we scan character is an ‘)’, we need to pop and print from the stack
        // do this until an ‘(‘ is encountered in the stack.
        else if (expression[i] == ')')
```

```
            {
                while (!isEmpty(stack) && peek(stack) != '(')
                    expression[++j] = pop(stack);
                if (!isEmpty(stack) && peek(stack) != '(')
                    return -1; // invalid expression
                else
                    pop(stack);
            }
            else // if an opertor
            {
                while (!isEmpty(stack) && precedence(expression[i]) <=
precedence(peek(stack)))
                    expression[++j] = pop(stack);
                push(expression[i]);
            }

        }

        // Once all inital expression characters are traversed
        // adding all left elements from stack to exp
        while (!isEmpty(stack))
            expression[++j] = pop(stack);

        expression[++j] = '\0';

    }
```

```
void reverse(char *exp){


    int size = strlen(exp);

    int j = size, i=0;

    char temp[size];


    temp[j--]='\0';

    while(exp[i]!='\0')

    {

        temp[j] = exp[i];

        j--;

        i++;

    }

    strcpy(exp,temp);

}

void brackets(char* exp){

    int i = 0;

    while(exp[i]!='\0')

    {

        if(exp[i]=='(')

            exp[i]=')';

        else if(exp[i]==')')

            exp[i]='(';

        i++;
    }
}
void InfixtoPrefix(char *exp){

    int size = strlen(exp);

    // reverse string
    reverse(exp);
    //change brackets
    brackets(exp);
    //get postfix
    getPostfix(exp);
    // reverse string again
    reverse(exp);
}

int main()
{
    printf("The infix is: ");

    char expression[] = "((a/b)+c)-(d+(e*f))";
    printf("%s\n",expression);
    InfixtoPrefix(expression);

    printf("The prefix is: ");
    printf("%s\n",expression);


    return 0;
  }
```

OUTPUT:



```
The infix is: ((a/b)+c)-(d+(e*f))
The prefix is: -+/ +d*ef
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 10

BINARY SEARCH

PROBLEM DEFINITION:

Write a program to implement Binary Search.

ALGORITHM:

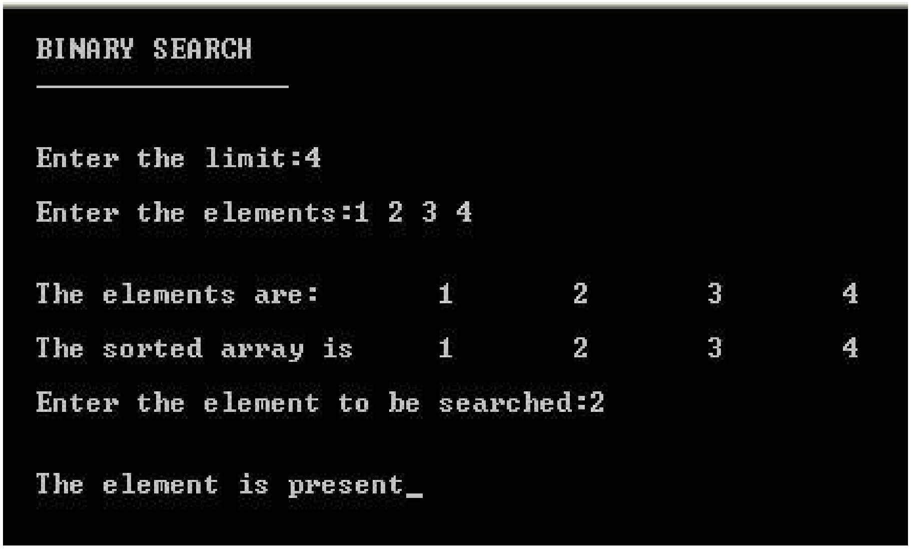
```
Step 1: Start
Step 2: Read the Array size
Step 3: Read the array elements
Step 4: Loop (i<n)
        Step 4.1: Read the array elements
Step 5: Loop ends
Step 6: Loop (i<n)
        Step 6.1: Loop (j<n-1)
        Step 6.2: If (a[i]>a[j+1])
                Step 6.2.1: Set temp as a[i]
                Step 6.2.2: Set a[i] as a[j+1]
                Step 6.2.3: Set a[j+1] as temp
        Step 6.3: Loop ends
Step 7: Loop ends
Step 8: Loop (i<n)
        Step 8.1: Print Sorted array
Step 9: Loop ends
Step10: Set begin as zero
Step11: Set end as n
Step 12: Print enter the key
Step 13: Read key
Step 14: Loop (begin <end)
        Step 14.1: Set mid as (begin+end)/2
        Step 14.2: If (a[mid]==key)
                Step 14.2.1: Print element key found at position mid
        Step 14.3: If ends
        Step 14.4: If (key>a[mid])
                Step 14.4.1: Set begin as mid +1
        Step 14.5: Endif
        Step 14.6: Else
Step 14.6.1: Set end as mid -1 Step 14.7: Else end
```

```
        Step 14.8: If (begin++ end)
                Step 14.8.1: Print not found
        Step 14.9: If ends
Step 15:
Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[50],i,j,n,elt,temp,flag=0,low=0,high,mid;
clrscr(); printf("\n\tBINARY
SEARCH");
printf("\n\t________________\n
\n");
printf("\n\tEnter the limit:");
scanf("%d",&n);
printf("\n\tEnter the
elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("\n\n\tThe elements are:");
for(i=0;i<n;i++)
{
printf("\t%d",a[
i]);
}
for(i=0;i<n-1;i++)
{
for(j=0;j<n-1-i;j++)
{
if(a[j]>a[j+
1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
printf("\n\n\tThe sorted array is");

for(i=0;i<n;i++)
{
printf("\t%d",a[
i]);
}
printf("\n\n\tEnter the element to be searched:");
scanf("%d",&elt);
high=n-1;
while(low<=high)
{
mid=(low+high)/2;
if(elt<a[mid])
{
high=mid-1;
}
else if(elt>a[mid])
{
low=mid+1;
}
else
{
printf("\n\n\tThe element is present");
flag=1;
break;
}
}
if(flag==0)
{
printf("\n\n\tThe element is not present");
}
getc
h();
}
                    break;
            }
    } while (ch != 'e');
    return 0;
```

OUTPUT:



```
BINARY SEARCH
_______________

Enter the limit:4

Enter the elements:1 2 3 4

The elements are:       1       2       3       4

The sorted array is     1       2       3       4

Enter the element to be searched:2

The element is present_
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 11

SINGLY LINKED LIST

PROBLEM DEFINITION:

Write a program to implement Singly Linked List.

ALGORITHM:

```
Step 1: Start
Step 2: Read the option
Step 3: If (traverse)
        Step 3.1: Ptr=header->link
        Step 3.2: While (ptr!=Null)do
                Step 3.2.1: Print ptr->data
                Step 3.2.2: Ptr=ptr->link
        Step 3.3: End while
        Step 3.4: Stop
Step 4: Elseif (insertion at front)
        Step 4.1: New = getnode (Node)
        Step 4.2: If (new=null)
                Step 4.2.1: Insertion not possible
                Step 4.2.2: Exit
        Step 4.3: Else
                Step 4.3.1: New->link=header->link
                Step 4.3.2: Header->link=new
        Step 4.4: Endif
        Step 4.5: Stop
Step 5: Elseif (insertion at end)
        Step 5.1: New = getnode (node)
        Step 5.2: If (new=null) then
                Step 5.2.1: Print insufficient memory
                Step 5.2.2: Exit
        Step 5.3: Else
                Step 5.3.1: Ptr=header
Step 5.3.2: While (ptr->link#null)
Step 5.3.2.1: Ptr=ptr->link
                Step 5.3.3: End while
                Step 5.3.4: Ptr->link=new
                Step 5.3.5: New->data=x
                Step 5.3.6: New->link=null
Step 5.4: Endif

        Step 5.5: Stop
Step 6: Else if (inset at any position)
```

```
        Step 6.1: New = getnode (node)
        Step 6.2: If (new==null)
                Step 6.2.1: Printf insufficient memory
                Step 6.2.2: Exit
        Step 6.3: Else
                Step 6.3.1: Ptr=header
                Step 6.3.2: While (ptr->data#key) and (ptr->link#null)
                        Step 6.3.2.1: Ptr=ptr->link
                Step 6.3.3: End while
                Step 6.3.4: If (ptr->link=null)
                        Step 6.3.4.1: Print key not available
                        Step 6.3.4.2: Exit
                Step 6.3.5: Else
                        Step 6.3.5.1: New->link=ptr->link
                        Step 6.3.5.2: New->data=v
                        Step 6.3.5.3: Ptr->link =new
                Step 6.3.6: Endif
        Step 6.4: Endif
        Step 6.5:  Stop
    Step 7: Elseif (deletefront)
        Step 7.1: Ptr=header->link
        Step 7.2: If (ptr=null) then
                Step 7.2.1: Print Empty list
                Step 7.2.2: Exit
        Step 7.3: Else
                Step 7.3.1: Ptr1=ptr->link
                Step 7.3.2: Header->link=ptr1
                Step 7.3.3: Return node (ptr)
        Step 7.4: Endif
        Step 7.5:  Stop
    Step 8: Else if (delete end)
        Step 8.1: Ptr=header->link
        Step 8.2: If (ptr=null) then
                Step 8.2.1: Print empty list
                Step 8.2.2: Exit
        Step 8.3: Else
                Step 8.3.1: While (ptr->link#Null)
                        Step 8.3.1.1: Ptr1=ptr
                        Step 8.3.1.2: Ptr=ptr->link
                Step 8.3.2: End while
                Step 8.3.3: Ptr->link=null
                Step 8.3.4: Return node(ptr);

                Step 8.3.5: Endif
                Step 8.3.6: Stop
Step 9: Elseif (delete any)
        Step 9.1: Ptr1=header
        Step 9.2: Ptr=ptr1->link
        Step 9.3: If (ptr==null) then
                Step 9.3.1: Print Empty list
                Step 9.3.2: Exit
        Step 9.4: Else
                Step 9.4.1: While (ptr#null) and (ptr->data#key) do
                        Step 9.4.1.1: Ptr1=ptr
                        Step 9.4.1.2: Ptr=ptr->link
                Step 9.4.2: End while
                Step 9.4.3: If (ptr==null) then
                        Step 9.4.3.1: Print Key not present
                        Step 9.4.3.2: Exit
                Step 9.4.4: Else
                        Step 9.4.4.1: Ptr1->link=ptr->link
                        Step 9.4.4.2: Return node (ptr)
                Step 9.4.5: End if
                Step 9.4.6: Stop
Step 9: Endif
Step 10:
Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
#include<alloc. h>
void traverse();
void insertfront();
void insertend();
void insertany();
void deletefront();
void deleteend();
void deleteany();

typedef struct node
{ int data;
struct node
*link;
}NODE;
NODE
*header=NULL,*newptr=NULL,*ptr,*ptr1;
```

```
void main()
{ int
no,ite
m;
char
c;
clrscr(
);
printf("\n\tPROGRAM TO PERFORM OPERATIONS ON SINGLE LINKED
LIST"); printf("\n\t         "); do
{
printf("\n\t\t\t\tMENU\n\n");
printf("\t\t1.TRAVERSE\n\t\t2.INSERT AT FRONT\n\t\t3.INSERT AT END\n\t\t4.INSERT
AT
ANY POSITION\n\t\t5.DELETE FROM FRONT\n\t\t6.DELETE FROM
END\n\t\t7.DELETE
FROM ANY POSITION\n\t\t8.EXIT\n\t\tEnter your
choice: "); scanf("%d",&no); if(no==8) break;
switch(no)
{
ca se 1:
        traverse();
    break;
case 2:

insertfront();
    break; case
3:
        insertend();
    break;
case 4:
        insertany();
    break;
case 5:
```

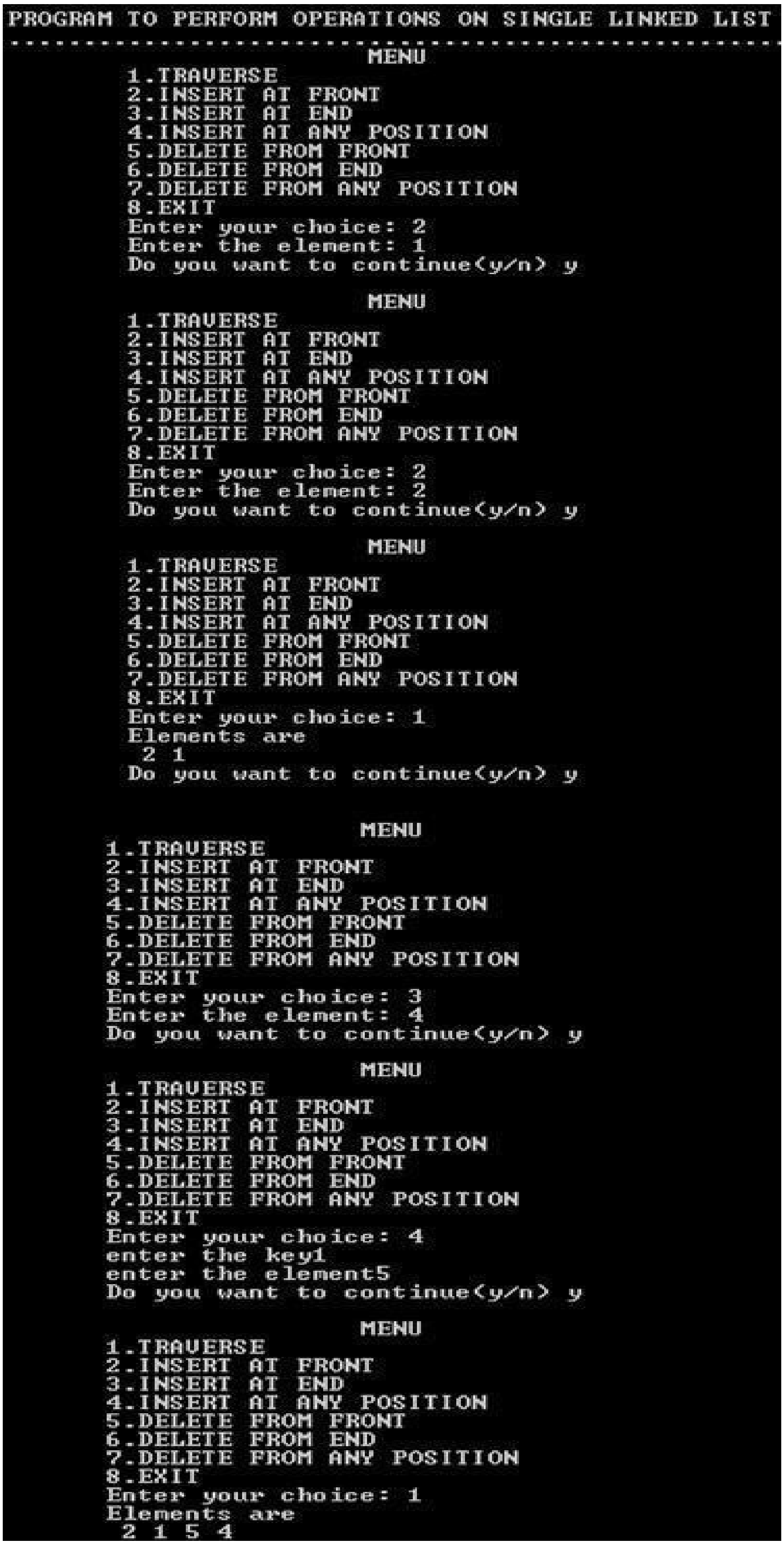
```
deletefront();
           break;
         case 6:
                    deleteend();
              break;
         case 7:
                    deleteany();
              break;
         default :
                   printf("\t\tINVALID ENTRY");
                   break;
         }
         printf("\t\tDo you want to continue(y/n) ");
         scanf(" %c",&c);
         }
         while(c=='y'||c=='Y'); getch(); } void insertfront()
         {
         newptr=(NODE*)malloc(sizeof(NODE));
         printf("\t\tEnter the element: ");
         scanf("%d",&newptr->data);
         newptr->link=NULL;
         if(newptr==NULL)
         printf("\t\tInsufficient memmory");
         else
         {
         newptr->link=header->link;
         header->link=newptr;
         }
         } void
         insertend()
         {
         newptr=(NODE*)malloc(sizeof(NODE));
         if(newptr==NULL)
         printf("\t\tInsufficient memmory");
         else
         {
         printf("\t\tEnter the element: "); scanf("%d",&newptr-
         >data); newptr->link=NULL;
```

```
ptr=header; while(ptr-
>link!=NULL) ptr=ptr->link; newptr->link=ptr->link;
ptr->link=newptr;
}
} void
insertany()
{
int key;
newptr=(NODE*)malloc(sizeof(NODE));
if(newptr==NULL)
printf("\t\tInsufficient memmory");
else {
printf("\t\tenter the key"); scanf("%d",&key); printf("\t\tenter the element");
scanf("%d",&newptr
->data);
ptr=header->link;
while(ptr-
>data!=key&&ptr!
=NULL)
ptr=ptr->link;
if(ptr==NULL)
printf("\t\tkey is not found");
else
{
newptr->link=ptr->link;
ptr->link=newptr;
}
} }
void deletefront()

{
ptr=header->link;
ptr1=ptr->link;
if(ptr==NULL)
printf("\t\tEmpty list");
else
{
header->link=ptr1; printf("\t\tdeleted
element is %d",ptr->data);
```

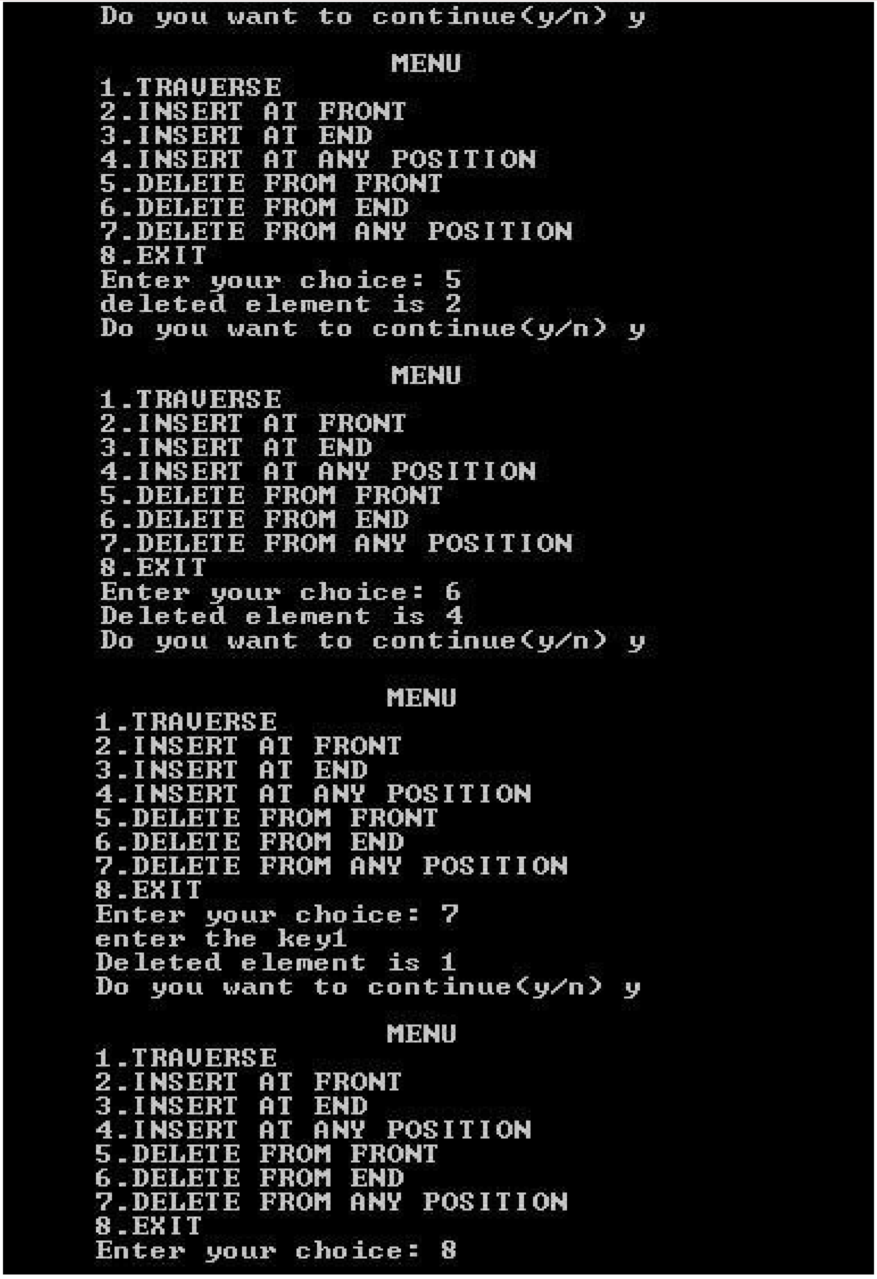
```
free(ptr); }
printf("\n")
; } void
deleteend()
{
ptr=header; ptr1=ptr-
>link; if(ptr1==NULL) printf("\t\tEmpty list");
 else
{
while(ptr1->link!=NULL)
{ ptr=ptr-
>link;
ptr1=ptr1->link;
}
ptr->link=NULL;
printf("\t\tDeleted    element    is    %d",ptr1-
>data); free(ptr1);
 }
 printf("\n");
}
void deleteany()
{ int key; ptr=heade r; ptr1=ptr-
>link; if(ptr1==NULL)
printf("\t\tEmpty list"); else {
printf("\t\tenter the key");
scanf("%d",&key); while(ptr1-
>data!=key&&ptr1!=NULL)
{ ptr=ptr1;
ptr1=ptr1-
>link;
}
if(ptr1==NULL)
printf("\t\tKey not found");
else if(ptr1->data==key)
{
ptr->link=ptr1->link;
printf("\t\tDeleted element is %d",ptr1-
>data);
```

```
 free(ptr1);
}
printf("\n");
 }}
void traverse()
{
if(header->link==NULL)
printf("\t\tlist is
empty\n"); else {
printf("\t\tElements are\n");
ptr=header->link;
printf("\t\t");
while(ptr!=NULL)
{ printf(" %d",ptr-
>data); ptr=ptr-
>link;
}
printf("\n")
;
}
}
```

OUTPUT:



```
PROGRAM TO PERFORM OPERATIONS ON SINGLE LINKED LIST
-----------------------------------------------------
                        MENU
        1.TRAVERSE
        2.INSERT AT FRONT
        3.INSERT AT END
        4.INSERT AT ANY POSITION
        5.DELETE FROM FRONT
        6.DELETE FROM END
        7.DELETE FROM ANY POSITION
        8.EXIT
        Enter your choice: 2
        Enter the element: 1
        Do you want to continue(y/n) y

                        MENU
        1.TRAVERSE
        2.INSERT AT FRONT
        3.INSERT AT END
        4.INSERT AT ANY POSITION
        5.DELETE FROM FRONT
        6.DELETE FROM END
        7.DELETE FROM ANY POSITION
        8.EXIT
        Enter your choice: 2
        Enter the element: 2
        Do you want to continue(y/n) y

                        MENU
        1.TRAVERSE
        2.INSERT AT FRONT
        3.INSERT AT END
        4.INSERT AT ANY POSITION
        5.DELETE FROM FRONT
        6.DELETE FROM END
        7.DELETE FROM ANY POSITION
        8.EXIT
        Enter your choice: 1
        Elements are
         2 1
        Do you want to continue(y/n) y


                       MENU
       1.TRAVERSE
       2.INSERT AT FRONT
       3.INSERT AT END
       4.INSERT AT ANY POSITION
       5.DELETE FROM FRONT
       6.DELETE FROM END
       7.DELETE FROM ANY POSITION
       8.EXIT
       Enter your choice: 3
       Enter the element: 4
       Do you want to continue(y/n) y
                       MENU
       1.TRAVERSE
       2.INSERT AT FRONT
       3.INSERT AT END
       4.INSERT AT ANY POSITION
       5.DELETE FROM FRONT
       6.DELETE FROM END
       7.DELETE FROM ANY POSITION
       8.EXIT
       Enter your choice: 4
       enter the key1
       enter the element5
       Do you want to continue(y/n) y
                       MENU
       1.TRAVERSE
       2.INSERT AT FRONT
       3.INSERT AT END
       4.INSERT AT ANY POSITION
       5.DELETE FROM FRONT
       6.DELETE FROM END
       7.DELETE FROM ANY POSITION
       8.EXIT
       Enter your choice: 1
       Elements are
        2 1 5 4
```



```
Do you want to continue(y/n) y

                MENU
1.TRAVERSE
2.INSERT AT FRONT
3.INSERT AT END
4.INSERT AT ANY POSITION
5.DELETE FROM FRONT
6.DELETE FROM END
7.DELETE FROM ANY POSITION
8.EXIT
Enter your choice: 5
deleted element is 2
Do you want to continue(y/n) y

                MENU
1.TRAVERSE
2.INSERT AT FRONT
3.INSERT AT END
4.INSERT AT ANY POSITION
5.DELETE FROM FRONT
6.DELETE FROM END
7.DELETE FROM ANY POSITION
8.EXIT
Enter your choice: 6
Deleted element is 4
Do you want to continue(y/n) y

                MENU
1.TRAVERSE
2.INSERT AT FRONT
3.INSERT AT END
4.INSERT AT ANY POSITION
5.DELETE FROM FRONT
6.DELETE FROM END
7.DELETE FROM ANY POSITION
8.EXIT
Enter your choice: 7
enter the key1
Deleted element is 1
Do you want to continue(y/n) y

                MENU
1.TRAVERSE
2.INSERT AT FRONT
3.INSERT AT END
4.INSERT AT ANY POSITION
5.DELETE FROM FRONT
6.DELETE FROM END
7.DELETE FROM ANY POSITION
8.EXIT
Enter your choice: 8
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

EXPERIMENT NO: 12

POLYNOMIAL USING LINKED LIST

AIM :

To write program to implement Polynomial using Linked List..

ALGORITHM :

```
Step 1: Start
Step 2: Read coefficient &exponent of 2 polynomials
Step 3: Compare the exponents from 1st node
Step 4: If (poly1->coeff>poly2->coeff)
    Step 4.1: Poly->coeff=poly1->coeff
    Step 4.2: Poly->exp=poly1->exp
    Step 4.3: Poly1=poly1->link
    Step 4.4: Poly->link = getnode (node)
    Step 4.5: Poly=poly->link
    Step 4.6: Exit
Step 5: Elseif (poly->exp<poly2->exp)
    Step 5.1: Poly->coeff=poly2->coeff
    Step 5.2: Poly->exp=poly2->exp
    Step 5.3:Poly2=poly2->link
    Step 5.4: Poly->link = getnode (node)
    Step 5.5: Poly=poly->link
    Step 5.6: Exit
Step 6: Else
    Step 6.1: Poly->coeff=poly2->coeff+poly1->coeff
    Step 6.2: Poly->exp=poly1->exp
    Step 6.3: Poly2=poly2->link
    Step 6.4: Poly1=poly1->link
    Step 6.5: Poly->link = getnode (node)
    Step 6.6: Poly=poly->link
    Step 6.7: Exit
Step 7: If (poly1->link!=Null)
    Step 7.1: While (poly1->link!=Null)
        Step 7.1.1: Poly->coeff=poly1->coeff
        Step 7.1.2: Poly->exp=poly1->exp
        Step 7.1.3: Poly1=poly1->link
        Step 7.1.4: Poly->link=getnode (node)
        Step 7.1.5: Poly=polylink
    Step 7.2: End while
Step 8: Else if
    Step 8.1: While (poly2link!=Null)
        Step 8.1.1: Poly->coeff=poly2->coeff
        Step 8.1.2: Poly->exp=poly2->exp
```

```
                    Step 8.1.3: Poly2=poly2->link
Step 8.1.4: Poly->link = getnode (node)
                    Step 8.1.5: Poly=poly->link
            Step 8.2: End while
    Step 9: End if
    Step 10: Stop
```

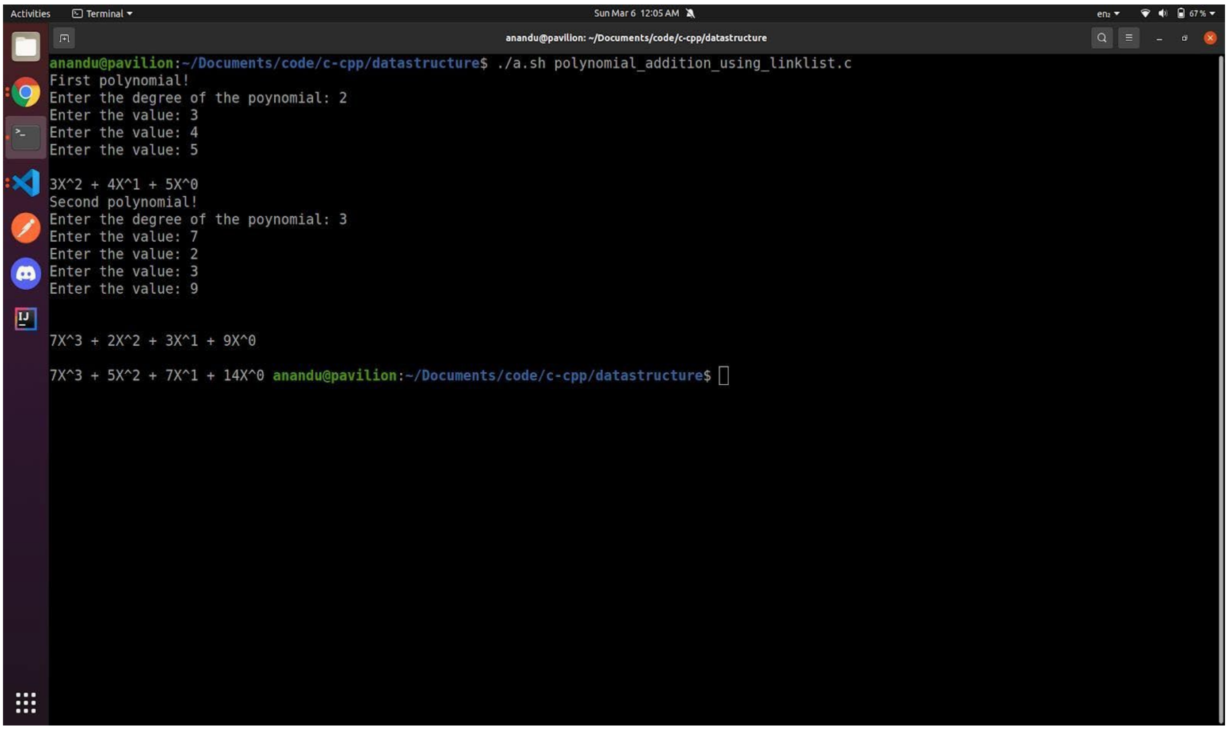
PROGRAM CODE:

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

typedef struct poly1
{
  int coeff;
  int exp;
  struct poly1 *next;
} poly;

void display(poly *header);
poly *Qheader, *Pheader, *Rheader;

poly *getnode()
{
  poly *temp = (poly *)malloc(sizeof(poly));
  temp->coeff = 0;
  temp->exp = 0;
  temp->next = NULL;
  return temp;
}

poly *Createpolynomial()
{
  printf("Enter the degree of the poynomial: ");
  int n;
  scanf("%d", &n);
  poly *header = getnode();
  poly *ptr1 = header;
  for (int i = n; i >= 0; i--)
  {
    printf("Enter the value: ");
    int x;
    scanf("%d", &x);
    if (x == 0)
    continue;
    poly *temp = getnode();
    temp->coeff = x;
    temp->exp = i;
    ptr1->next = temp;
    ptr1 = ptr1->next;
  }
```

```
return header;
}



void polyadd()
{
 Rheader = getnode();
 poly *Rptr = Rheader;
 poly *Pptr = Pheader->next;
 poly *Qptr = Qheader->next;
 while (Pptr != NULL && Qptr != NULL)
 {
  if (Pptr->exp == Qptr->exp)
  {
   poly *temp = getnode();
   temp->exp = Pptr->exp;
   temp->coeff = Pptr->coeff + Qptr->coeff;
   Pptr = Pptr->next;
   Qptr  =  Qptr->next;
   Rptr->next = temp;
   Rptr = Rptr->next;
  }
  else if (Pptr->exp > Qptr->exp)
  {
   poly *temp = getnode();
   temp->exp = Pptr->exp;
   temp->coeff = Pptr->coeff;
   Pptr = Pptr->next;
   Rptr->next = temp;
   Rptr = Rptr->next;
  }
  else if (Pptr->exp < Qptr->exp)
  {
   poly *temp = getnode();
   temp->exp = Qptr->exp;
   temp->coeff = Qptr->coeff;
   Qptr = Qptr->next;
   Rptr->next = temp;
   Rptr = Rptr->next;
  }
 }
 while (Pptr != NULL)
 {
  poly *temp = getnode();
  temp->exp = Pptr->exp;
  temp->coeff = Pptr->coeff;
  Pptr = Pptr->next;
  Rptr->next = temp;
  Rptr = Rptr->next;
 }
 while (Qptr != NULL)
 {
```

```
poly *temp = getnode();
  temp->exp = Qptr->exp;
  temp->coeff = Qptr->coeff;
  Qptr = Qptr->next;
  Rptr->next = temp;
  Rptr = Rptr->next;
 }
}

void display(poly *header)
{
 printf("\n");
 poly *ptr = header->next;
 while (ptr != NULL)
 {
  if (ptr->next != NULL)
  {
   printf("%dX^%d + ", ptr->coeff, ptr->exp);
  }
  else
  {
   printf("%dX^%d ", ptr->coeff, ptr->exp);
  }
  ptr = ptr->next;
 }
}

int main()
{
 printf("First polynomial!\n");
 Pheader = Createpolynomial();
 display(Pheader);
 printf("\n");
 printf("Second polynomial!\n");
 Qheader  =  Createpolynomial();
 printf("\n");
 display(Qheader);
 printf("\n");
 polyadd();
 display(Rheader);
}
```

OUTPUT:



```
anandu@pavilion:~/Documents/code/c-cpp/datastructure$ ./a.sh polynomial_addition_using_linklist.c
First polynomial!
Enter the degree of the poynomial: 2
Enter the value: 3
Enter the value: 4
Enter the value: 5

3X^2 + 4X^1 + 5X^0
Second polynomial!
Enter the degree of the poynomial: 3
Enter the value: 7
Enter the value: 2
Enter the value: 3
Enter the value: 9


7X^3 + 2X^2 + 3X^1 + 9X^0

7X^3 + 5X^2 + 7X^1 + 14X^0 anandu@pavilion:~/Documents/code/c-cpp/datastructure$ 
```

RESULT :

Result has been obtained and the output has been verified.

Experiment No: 13

INSERTION SORT

PROBLEM DEFINITION:

Write a program to implement Insertion Sort.

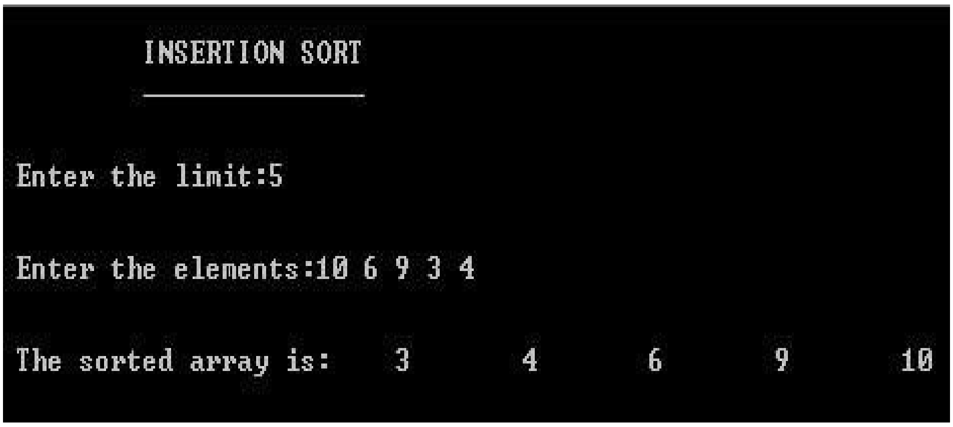
ALGORITHM:

```
Step 1: Start
Step 2: Read the array size
Step 3: Read the elements
Step 4: Loop (i<n)
        Step 4.1: Read array elements
Step 5: Loop ends
Step 6: Set I to 1
Step 7: Loop (i<n)
        Step 7.1: Set temp as a[i]
        Step 7.2: Set j as i-1
        Step 7.3: Loop (a[i]>temp & j>=0)
                Step 7.3.1: Set a[j+1]=a[j]
                Step 7.3.2: Decrement j
        Step 7.4: End loop
        Step 7.5: Set a[j+1] as temp
Step 8: End loop
Step 9: Loop (i<n)
        Step 9.1: Print the sorted array
Step10: Loop ends
Step 11: Stop
```

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
#include<conio.h>
void main()
{ int
a[50],i,n,j,tem
p;
clrscr();
printf("\n\t\tINSERTION SORT\n");
printf("\t\t______________\n");
printf("\n\n\tEnter the limit:"); scanf("%d",&n);
printf("\n\n\tEnter the elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for(i=1;i<n;i++)
{
temp=a[i];
j=i-1;
while(temp<a[j]&&j>=0)
{
a[j+1]=a[j];
 j--;
 }
a[j+1]=temp;
}
printf("\n\n\tThe sorted array is:");
for(i=0;i<n;i++)
{
printf("\t%d",a[i]);
}
getc h();
}
```

OUTPUT:



```
        INSERTION SORT
        ______________

Enter the limit:5

Enter the elements:10 6 9 3 4

The sorted array is:    3       4       6       9       10
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 14

SELECTION SORT

PROBLEM DEFINITION:

Write a program to implement Selection Sort.

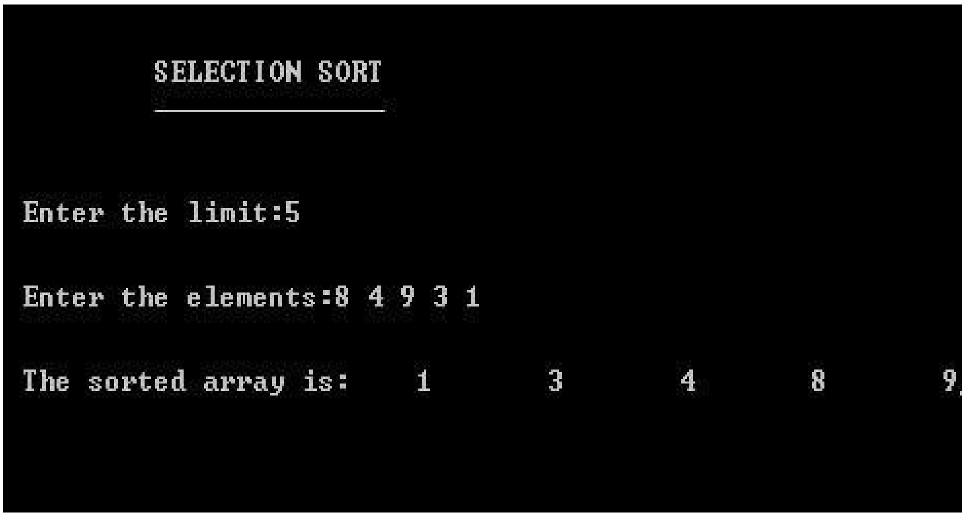
ALGORITHM:

```
Step 1: Start
Step 2: Read the size of the array
Step 3: Read the elements
Step 4: Set I to 0
Step 5: Loop (i<n)
        Step 5.1: Read elements
        Step 5.2: Increment i
Step 6: Loop ends
Step 7: Loop (i<n)
        Step 7.1: Set j=i+1
        Step 7.2: Loop (j<n)
                Step 7.2.1: If (a[j]>a[i])
                        Step 7.2.1.1: Set temp=a[i]
                        Step 7.2.1.2: Set a[i]=a[j]
                        Step 7.2.1.3: Set a[j]=temp
                Step 7.2.2:  Endif
                Step 7.2.3: Loop ends
Step 8: Loop ends
Step 9: Loop (i<n)
        Step 9.1: Print the sorted array
Step 10: Loop ends
Step 11: Stop
```

PROGRAM DEVELOPMENT:

```
    #include<stdio.h>
    #include<conio.h>
    void main()
    { int a[50],i,n,j,temp;
clrscr();
    printf("\n\n\t\tSELECTION SORT"); printf("\n\t\t____\n\n");
    printf("\n\n\tEnter the limit:"); scanf("%d",&n);
    printf("\n\n\tEnter the elements:");
    for(i=0;i<n;i++)
    {
    scanf("%d",&a[i]);
    }
    for(i=0;i<n-1;i++)
    {
    for(j=i+1;j<n;j++)
    {
    if(a[i]>a[j])
    {
    temp=a[i];
    a[i]=a[j];
    a[j]=temp;
    }
    }
    }
    printf("\n\n\tThe sorted array is:");
    for(i=0;i<n;i++)
    {
    printf("\t%d",a[i]);
    }
    getch();
    }
```

OUTPUT:



```
        SELECTION SORT
        ______________


Enter the limit:5

Enter the elements:8 4 9 3 1

The sorted array is:    1       3       4       8       9
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 15

QUICK SORT

PROBLEM DEFINITION:

Write a program to implement Quick Sort.

ALGORITHM:

Step 1: Start

Step 2: Read the elements to be sort

Step 3: Find the proper pivot element

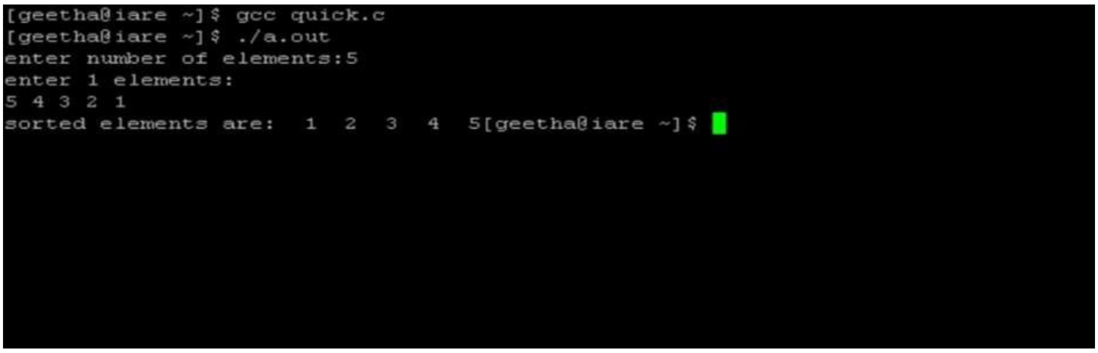
Step 4: Apply quick sort method to sort the remaining elements

PROGRAM DEVELOPMENT:

```
#include<stdio.h>
 void main()
{
 int x[10],i,n;
 printf("enter number of elements:");
 scanf("%d",&n);
 printf("enter %d elements:\n");
 for(i=0;i<n;i++)
 scanf("%d",&x[i]);
 quicksort(x,0,n-1);/*function call*/
 printf("sorted elements are:");
 for(i=0;i<n;i++)
 printf("%3d",x[i]);
}
/*called function*/
quicksort(int x[10],int first,int last)
{
 int pivot,i,j,t;
 if(first<last)
 {
 pivot=first;
 i=first;
```

```
        j=last;
         while(i<j)
         {
         while(x[i]<=x[pivot]&&i<last)
         i++;
         while(x[j]>x[pivot])
        j--;
         if(i<j)


        {
         t=x[i];
         x[i]=x[j];
         x[j]=t;

        }


        }
         t=x[pivot]; x[pivot]=x[j]; x[j]=t; quicksort(x,first,j
        -1);
         quicksort(x,j+1,last);
        }
}
```

OUTPUT:



```
[geetha@iare ~]$ gcc quick.c
[geetha@iare ~]$ ./a.out
enter number of elements:5
enter 1 elements:
5 4 3 2 1
sorted elements are:  1   2   3   4   5[geetha@iare ~]$ 
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.

Experiment No: 16

MERGE SORT

PROBLEM DEFINITION:

Write a program to implement Merge Sort.

ALGORITHM:

Step 1: Start

Step 2: If a given array A has zero or one element, simply return; it is already sorted. Otherwise, split A [p . r] into two subarrays A[p .. q] and A[q + 1 .. r], each containing about half of the elements of A[p .. r]. That is, q is the halfway point of A[p .. r].

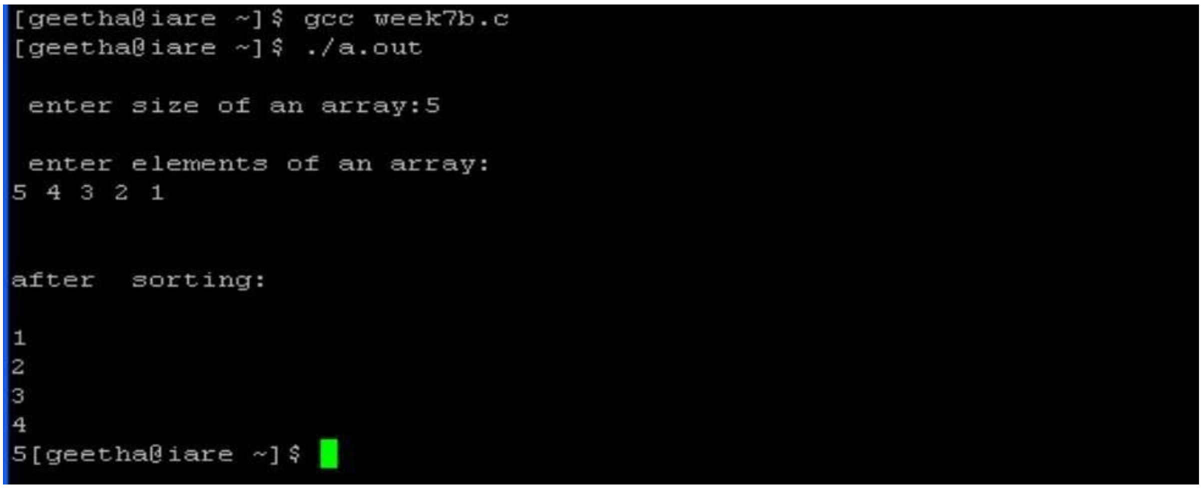
Step 3: Conquer by recursively sorting the two subarrays A[p .. q] and A[q + 1 .. r]. 3. Combine Step Combine the elements back in A[p .. r] by merging the two sorted subarrays A[p .. q] and A[q + 1 .. r] into a sorted sequence. To accomplish this step, we will define a procedure MERGE (A, p, q, r). 7.

PROGRAM DEVELOPMENT:

```
#include<stdio.h>

void mergesort(int[],int,int); void

mergearray(int[],int,int,int);

main()

{

 int a[50],n,i;

 printf("

\n enter size of an array:");

 scanf("%d",&n);

 printf("

\n enter elements of an array:

\n"); for(i=0;i<n;i++) scanf("%d",&a[i]);

mergesort(a,0,n

-1);

printf("\n

\nafter sorting:

\n");
```

```
 for(i=0;i<n;i++)
 printf("
\n%d",a[i]);
}
/*merge operation*/
void mergesort(int a[],int beg,int end)
{
 int mid;
 if(beg<end)
{
 mid=(beg+end)/2;
 mergesort(a,beg,mid);
 mergesort(a,mid+1,end);
 mergearray(a,beg,mid,end); } }
void mergearray(int a[],int beg,int mid,int end)
{
 int i,leftend,num,temp,j,k,b[50];
 for(i=beg;i<=end;i++)
 b[i]=a[i];
 i=beg;
 j=mid+1;
 k=beg;
 while((i<=mid)&&(j<=end))
{
 if(b[i]<+b[j])


 {
 a[k]=b[i];
 i++;
 k++;
}
 else
{
 a[k]=b[j];
j++;
 k++;

}
}
if(i<=mid)
{
 while(i<=mid)
{
 a[k]=b[i];
 i++;
 k++;

}
}
else
{
 while(j<=end)
{
31
 a[k]=b[j];
j++;


 k++;
 } } }
```

OUTPUT:



```
[geetha@iare ~]$ gcc week7b.c
[geetha@iare ~]$ ./a.out

 enter size of an array:5

 enter elements of an array:
5 4 3 2 1


after  sorting:

1
2
3
4
5[geetha@iare ~]$
```

CONCLUSION:

The algorithm was developed and the program was coded. The program was tested successfully.