NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE

(NAAC Accredited)

(Approved by AICTE , Affiliated to APJ Abdul Kalam Technological University, Kerala)

Pampady, Thiruvilwamala(PO), Thrissur(DT), Kerala 680 588

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

CSL201 DATA STRUCTURES LAB

VISION OF THE INSTITUTION

To mould true citizens who are millennium leaders and catalysts of change through excellence in education.

MISSION OF THE INSTITUTION

NCERC is committed to transform itself into a center of excellence in Learning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values.

We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

ABOUT THE DEPARTMENT

- ♦ Established in: 2002
- ♦ Course offered : B.Tech in Computer Science and Engineering

  M.Tech in Computer Science and Engineering

  M.Tech in Cyber Security
- ♦ Approved by AICTE New Delhi and Accredited by NAAC
- ♦ Certified by ISO 9001-2015
- ♦ Affiliated to A P J Abdul Kalam Technological University, Kerala.

DEPARTMENT VISION

Producing Highly Competent, Innovative and Ethical Computer Science and Engineering Professionals to facilitate continuous technological advancement.

DEPARTMENT MISSION

1. To Impart Quality Education by creative Teaching Learning Process
2. To Promote cutting-edge Research and Development Process to solve real world problems with emerging technologies.
3. To Inculcate Entrepreneurship Skills among Students.
4. To cultivate Moral and Ethical Values in their Profession.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1: Graduates will be able to Work and Contribute in the domains of Computer Science and Engineering through lifelong learning.
PEO2: Graduates will be able to analyze, design and development of novel Software Packages, Web Services, System Tools and Components as per needs and specifications.
PEO3: Graduates will be able to demonstrate their ability to adapt to a rapidly changing environment by learning and applying new technologies.
PEO4: Graduates will be able to adopt ethical attitudes, exhibit effective communication skills, Team work and leadership qualities.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

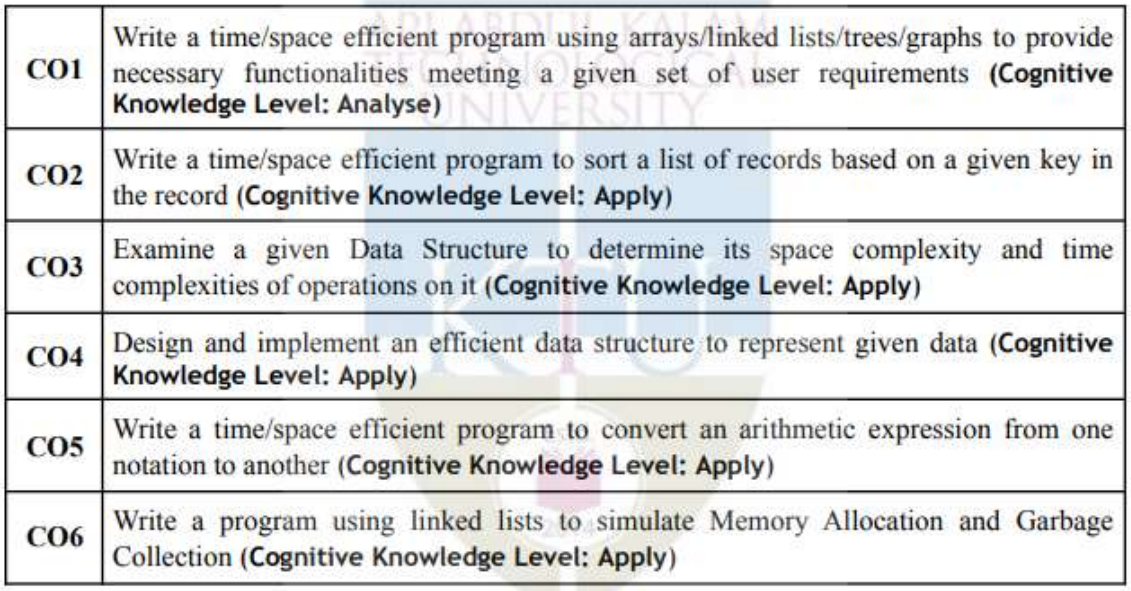
PROGRAM SPECIFIC OUTCOMES (PSO)

PSO1: Ability to Formulate and Simulate Innovative Ideas to provide software solutions for Real-time Problems and to investigate for its future scope.

PSO2: Ability to learn and apply various methodologies for facilitating development of high quality System Software Tools and Efficient Web Design Models with a focus on performance optimization.

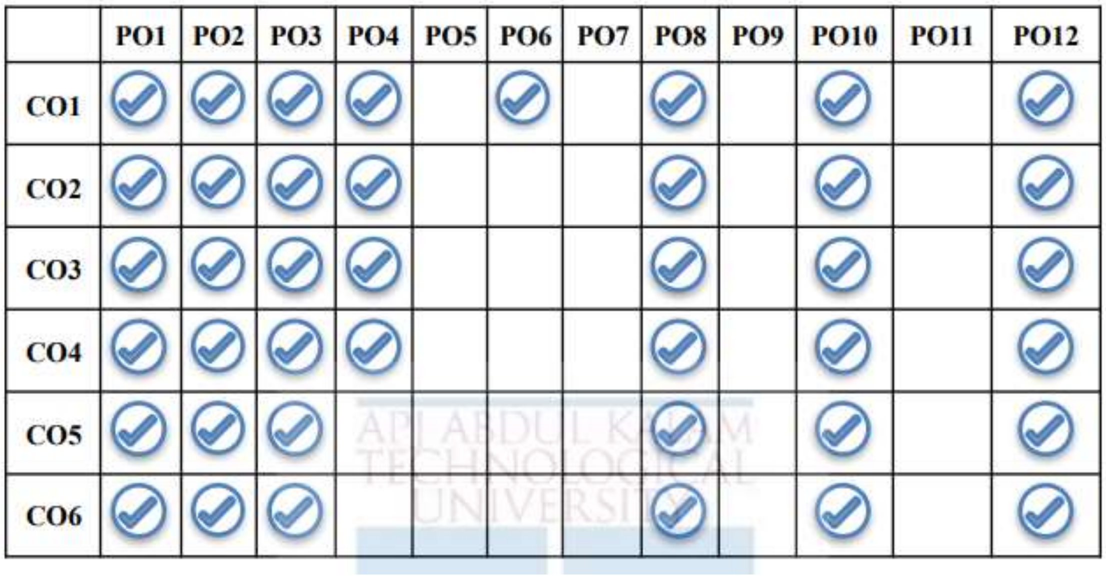
PSO3: Ability to inculcate the Knowledge for developing Codes and integrating hardware/software products in the domains of Big Data Analytics, Web Applications and Mobile Apps to create innovative career path and for the socially relevant issues.

COURSE OUTCOME



CO1	Write a time/space efficient program using arrays/linked lists/trees/graphs to provide necessary functionalities meeting a given set of user requirements **(Cognitive Knowledge Level: Analyse)**
CO2	Write a time/space efficient program to sort a list of records based on a given key in the record **(Cognitive Knowledge Level: Apply)**
CO3	Examine a given Data Structure to determine its space complexity and time complexities of operations on it **(Cognitive Knowledge Level: Apply)**
CO4	Design and implement an efficient data structure to represent given data **(Cognitive Knowledge Level: Apply)**
CO5	Write a time/space efficient program to convert an arithmetic expression from one notation to another **(Cognitive Knowledge Level: Apply)**
CO6	Write a program using linked lists to simulate Memory Allocation and Garbage Collection **(Cognitive Knowledge Level: Apply)**

MAPPING OF COURSE OUTCOMES WITH PROGRAM OUTCOMES



	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	✓	✓	✓	✓		✓		✓		✓		✓
CO2	✓	✓	✓	✓				✓		✓		✓
CO3	✓	✓	✓	✓				✓		✓		✓
CO4	✓	✓	✓	✓				✓		✓		✓
CO5	✓	✓	✓					✓		✓		✓
CO6	✓	✓	✓					✓		✓		✓

MAPPING OF COURSE OUTCOMES WITH PROGRAM SPECIFIC OUTCOMES

	PSO1	PSO2	PSO3
CO1	3	3	-
CO2	3	3	-
CO3	3	3	-
CO4	3	3	-
CO5	3	3	-
CO6	3	3	-

Note: H-Highly correlated=3, M-Medium correlated=2, L-Less correlated=1

PREPARATION FOR THE LABORATORY SESSION

GENERAL INSTRUCTIONS TO STUDENTS

1. Read carefully and understand the description of the experiment in the lab manual. You may go to the lab at an earlier date to look at the experimental facility and understand it better. Consult the appropriate references to be completely familiar with the concepts and hardware.

2. Make sure that your observation for previous week experiment is evaluated by the faculty member and you have transferred all the contents to your record before entering to the lab/workshop.

3. At the beginning of the class, if the faculty or the instructor finds that a student is not adequately prepared, they will be marked as absent and not be allowed to perform the experiment.

4. Bring necessary material needed (writing materials, graphs, calculators, etc.) to perform the required preliminary analysis. It is a good idea to do sample calculations and as much of the analysis as possible during the session. Faculty help will be available. Errors in the procedure may thus be easily detected and rectified.

5. Please actively participate in class and don’t hesitate to ask questions. Please utilize the teaching assistants fully. To encourage you to be prepared and to read the lab manual before coming to the laboratory, unannounced questions may be asked at any time during the lab.

6. Carelessness in personal conduct or in handling equipment may result in serious injury to the individual or the equipment. Do not run near moving machinery/equipment. Always be on the alert for strange sounds. Guard against entangling clothes in moving parts of machinery.

7. Students must follow the proper dress code inside the laboratory. To protect clothing from dirt, wear a lab coat. Long hair should be tied back. Shoes covering the whole foot will have to be worn.

8. In performing the experiments, please proceed carefully to minimize any water spills, especially on the electric circuits and wire.

9. Maintain silence, order and discipline inside the lab. Don’t use cell phones inside the laboratory.

10. Any injury no matter how small must be reported to the instructor immediately.

11. Check with faculty members one week before the experiment to make sure that you have the handout for that experiment and all the apparatus.

AFTER THE LABORATORY SESSION

1. Clean up your work area.

2. Check with the technician before you leave.

3. Make sure you understand what kind of report is to be prepared and due submission of record is next lab class.

4. Do sample calculations and some preliminary work to verify that the experiment was successful

MAKE-UPS AND LATE WORK

Students must participate in all laboratory exercises as scheduled. They must obtain permission from the faculty member for absence, which would be granted only under justifiable circumstances. In such an event, a student must make arrangements for a make-up laboratory, which will be scheduled when the time is available after completing one cycle. Late submission will be awarded less mark for record and internals and zero in worst cases.

LABORATORY POLICIES

1. Food, beverages & mobile phones are not allowed in the laboratory at any time.

2. Do not sit or place anything on instrument benches.

3. Organizing laboratory experiments requires the help of laboratory technicians and staff. Be punctual.

SYLLABUS

1. Implementation of Polynomials and Sparse matrices using arrays**
2. Implementation of Stack , Queues, Priority Queues, DEQUEUE and Circular Queues using arrays**
3. Application problems using stacks: Conversion of expression from one notation to another notation . **
4. Implementation of various linked list operations. **
5. Implementation of stack, queue and their applications using linked list.pression
6. Implementation of trees using linked list
7. Representation of polynomials using linked list, addition and multiplication of polynomials. **
8. Implementation of binary trees using linked lists and arrays- creations, insertion, deletion and traversal. **
9. Implementation of binary search trees – creation, insertion, deletion, search
10. Any application programs using trees
11. Implementation of sorting algorithms – bubble, insertion, selection, quick, merge sort

    and heap sort.**
12. Implementation of searching algorithms – linear search, binary search.**
13. Representation of graphs and computing various parameters (in degree, out degree etc.) - adjacency list, adjacency matrix.
14. Implementation of BFS and DFS for each graph representations.**
15. Implementation of hash table using your own mapping functions and observe collisions and overflow resolving schemes.**
16. Simulation of first-fit, best-fit and worst-fit allocations.
17. Simulation of a basic memory allocator and garbage collector using doubly linked list.
    ** mandatory.

INDEX

EXPERIMENT – 1
STACK USING ARRAYS

AIM

Write a program to implement Stack using arrays

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
```

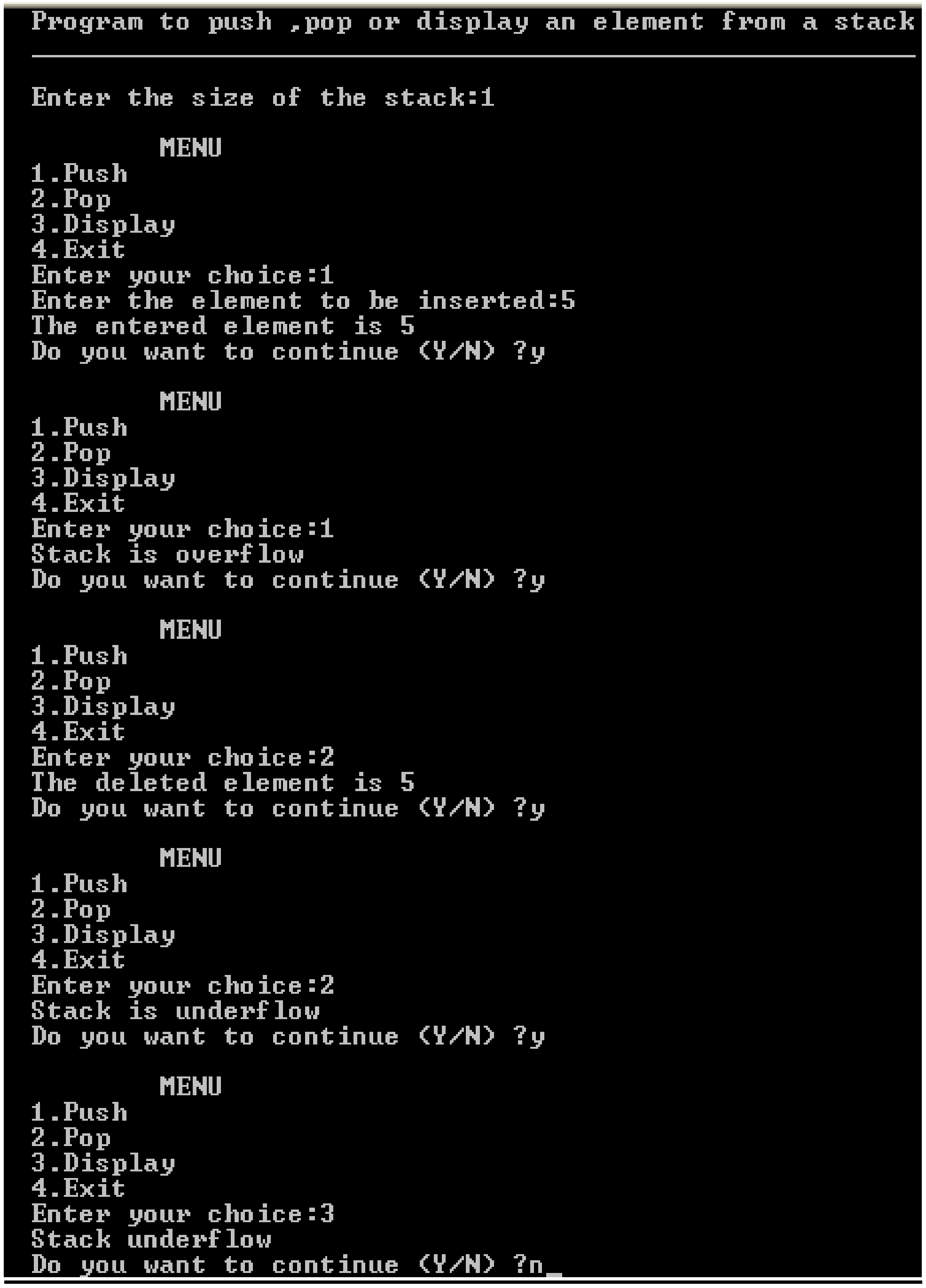
PROGRAM

```
#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_
```

RESULT AND DISCUSSION

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

VIVA QUESTIONS

1. What is a linear data structure?
2. What are the operations of stack?

3. What are the applications of Stack?
4. What is top value in stack?
5. How can a stack can be implemented?

EXPERIMENT – 2
QUEUE USING ARRAYS

AIM

Write a program to implement Queue.

ALGORITHM

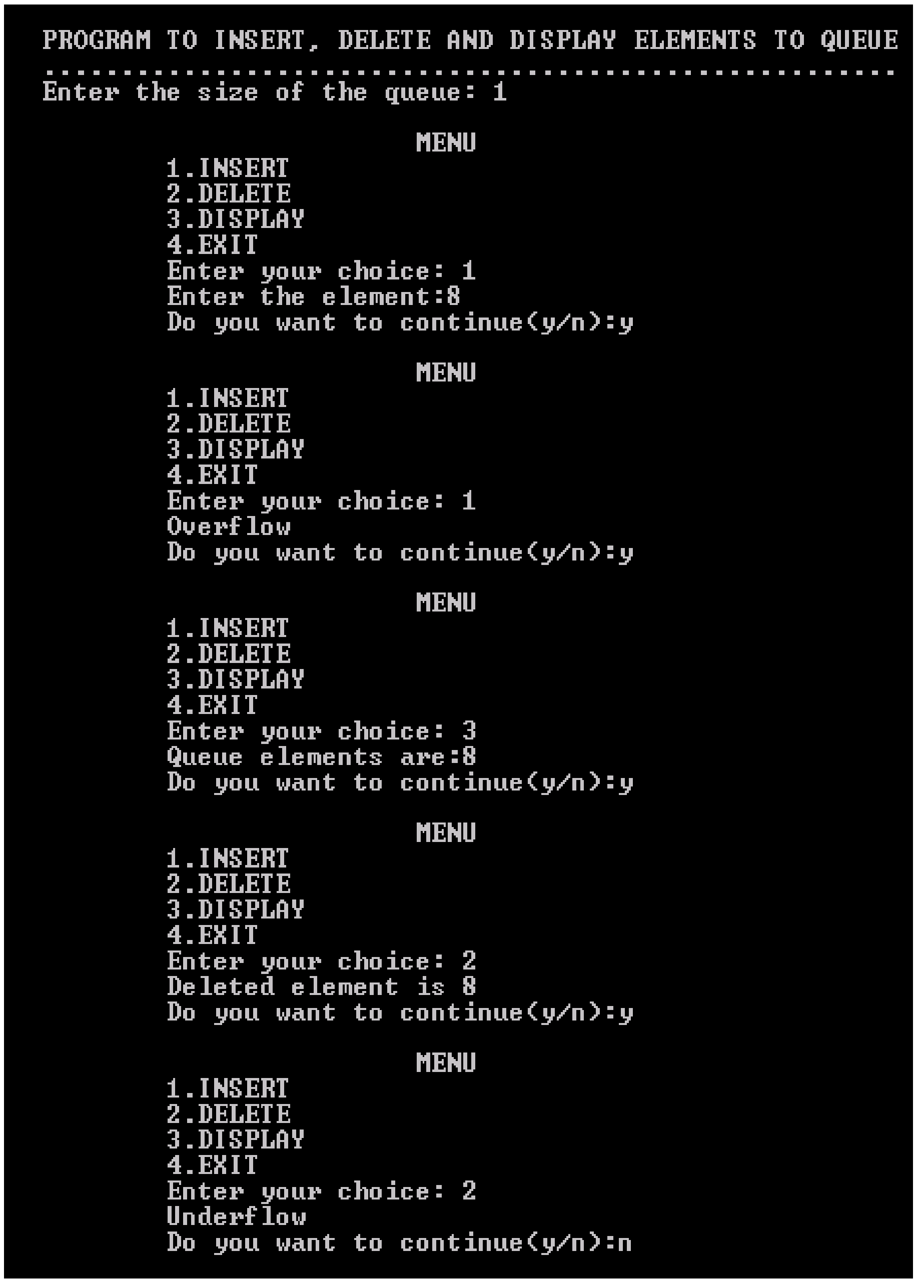
```
Step 1: Start
Step 2: Set front and rear to -1
Step 3: Read size of queue
Step 4: If (insertion)
    Step 4.1: If (rear=size-1)
        Step 4.1.1: Print overflow
    Step 4.2: End if
    Step 4.3: Else
        Step 4.3.1: Print enter the element
        Step 4.3.2: Read the element
        Step 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

```
#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\tOverflow");
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
```

RESULT AND DISCUSSION

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

VIVA QUESTIONS

1. What is a queue?
2. What are the operations on queue?
3. What are the applications of queue?

4. What are rear and front in queue?
5. How can a queue be implemented?

EXPERIMENT – 3
POLYNOMIAL ADDITION USING ARRAYS

AIM

Write a program to implement polynomial addition using arrays

ALGORITHM

```
 Step 1. [Initialize segment variables]
    [Initialize Counter] Set i=0,j=0,k=0
 Step 2.  Repeat step 3 while i<t1 and j<t2
 Step 3. If p1[i].expo=p2[j].expo, then
                  p3[i].coeff=p1[i].coeff+p2[i].coeff
                  p3[k].expo=p1[i].expo
                  [Increase counter] Set i=i+1,j=j+1,k=k+1
                  else if p1[i].expo > p2[j].expo, then
                            p3[k].coeff=p1[i].coeff
                            p3[k].expo=p1[i].expo
                            [Increase counter] Set i=i+1,k=k+1
                  else
                            p3[k].coeff=p2[j].coeff
                            p3[k].expo=p2[j].expo
                  Set j=j+1,k=k+1
                  [End of If]
                  [End of loop]

Step 4.  Repeat while i<t1
```

```
                p3[k].coeff=p1[i].coeff
                p3[k].expo=p1[i].expo
                Set i=i+1,k=k+1
                [End of loop]

Step 5.  Repeat while j<t2
                p3[k].coeff=p2[j].coeff
                p3[k].expo=p2[j].expo
                Set j=j+1,k=k+1
                 [End of loop]
Step 6.  Return k
Step 7.  Exit
```

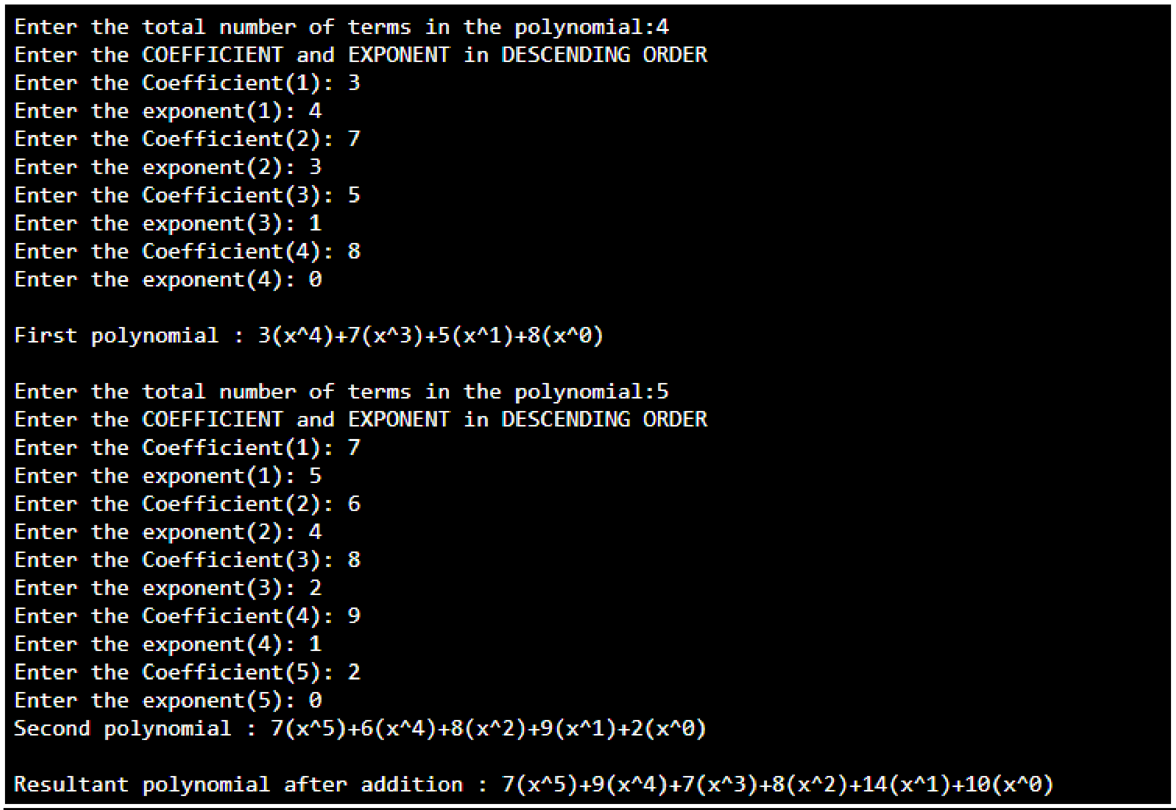
PROGRAM

```
 #include<stdio.h>
struct poly
 {
        int coeff;
        int expo;
 };
struct poly p1[10],p2[10],p3[10];
int readPoly(struct poly []);
 int addPoly(struct poly [],struct poly [],int ,int ,struct poly []);
 void displayPoly( struct poly [],int terms);


 int main()
```

```
{
        int t1,t2,t3;
        t1=readPoly(p1);
        printf(" \n First polynomial : ");
        displayPoly(p1,t1);
        t2=readPoly(p2);
        printf(" \n Second polynomial : ");
        displayPoly(p2,t2);
        t3=addPoly(p1,p2,t1,t2,p3);
        printf(" \n\n Resultant polynomial after addition : ");
        displayPoly(p3,t3);
        printf("\n");
        return 0;
}
int readPoly(struct poly p[10])
{
        int t1,i;
        printf("\n\n Enter the total number of terms in the polynomial:");
        scanf("%d",&t1);
        printf("\n Enter the COEFFICIENT and EXPONENT in DESCENDING
ORDER\n");
        for(i=0;i<t1;i++)
        {
                printf("   Enter the Coefficient(%d): ",i+1);
                scanf("%d",&p[i].coeff);
                printf("      Enter the exponent(%d): ",i+1);

                scanf("%d",&p[i].expo);        /* only statement in loop */

        }

        return(t1);

}


int addPoly(struct poly p1[10],struct poly p2[10],int t1,int t2,struct poly p3[10])

{

        int i,j,k;

        i=0;

        j=0;

        k=0;


        while(i<t1 && j<t2)

        {

                if(p1[i].expo==p2[j].expo)

                {

                        p3[k].coeff=p1[i].coeff + p2[j].coeff;

                        p3[k].expo=p1[i].expo;


                        i++;

                        j++;

                        k++;

                }

                else if(p1[i].expo>p2[j].expo)
        {
            p3[k].coeff=p1[i].coeff;
            p3[k].expo=p1[i].expo;
            i++;
            k++;
        }
        else
        {
            p3[k].coeff=p2[j].coeff;
            p3[k].expo=p2[j].expo;
            j++;
            k++;
        }
    }

    while(i<t1)
    {
        p3[k].coeff=p1[i].coeff;
        p3[k].expo=p1[i].expo;
        i++;
        k++;
    }
    while(j<t2)
    {
        p3[k].coeff=p2[j].coeff;
```

```
                p3[k].expo=p2[j].expo;
                j++;
                k++;
        }

        return(k); /* k is number of terms in resultant polynomial*/
 }
 void displayPoly(struct poly p[10],int term)
 {
        int k;
        for(k=0;k<term-1;k++)
        printf("%d(x^%d)+",p[k].coeff,p[k].expo);
        printf("%d(x^%d)",p[term-1].coeff,p[term-1].expo);
}
```

OUTPUT



```
Enter the total number of terms in the polynomial:4
Enter the COEFFICIENT and EXPONENT in DESCENDING ORDER
Enter the Coefficient(1): 3
Enter the exponent(1): 4
Enter the Coefficient(2): 7
Enter the exponent(2): 3
Enter the Coefficient(3): 5
Enter the exponent(3): 1
Enter the Coefficient(4): 8
Enter the exponent(4): 0

First polynomial : 3(x^4)+7(x^3)+5(x^1)+8(x^0)

Enter the total number of terms in the polynomial:5
Enter the COEFFICIENT and EXPONENT in DESCENDING ORDER
Enter the Coefficient(1): 7
Enter the exponent(1): 5
Enter the Coefficient(2): 6
Enter the exponent(2): 4
Enter the Coefficient(3): 8
Enter the exponent(3): 2
Enter the Coefficient(4): 9
Enter the exponent(4): 1
Enter the Coefficient(5): 2
Enter the exponent(5): 0
Second polynomial : 7(x^5)+6(x^4)+8(x^2)+9(x^1)+2(x^0)

Resultant polynomial after addition : 7(x^5)+9(x^4)+7(x^3)+8(x^2)+14(x^1)+10(x^0)
```

RESULT AND DISCUSSION

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

VIVA QUESTIONS

1. What is a polynomial?
2. How do you represent a polynomial?
3. What are the operations performed on polynomials?

EXPERIMENT – 4
SPARSE MATRIX

AIM

Write a program to implement sparse matrix using arrays.

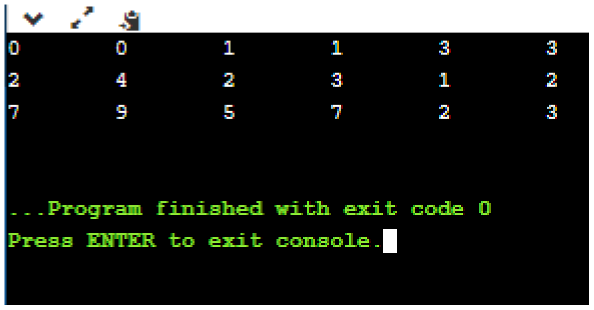
ALGORITHM

```
Step 1. Start.
Step 2. Declare the matrix sparse_matrix
Step 3. Determine the size of the matrix.
Step 4. for(int i=0; i<4; i++)  do repeat
            for(int j=0; j<5; j++)  do repeat
                  if sparse_matrix[i][j]!=0 then do
                 set   matrix[0][k] = i
                   matrix[1][k] = j
                   matrix[2][k] = sparse_matrix[i][j]
                   increment k
Step 5. Print matrix
Step 6. Stop
```

PROGRAM

```
#include <stdio.h>
int main()
{
   // Sparse matrix having size 4*5
   int sparse_matrix[4][5] =
   {
    {0 , 0 , 7 , 0 , 9 },
    {0 , 0 , 5 , 7 , 0 },
    {0 , 0 , 0 , 0 , 0 },
    {0 , 2 , 3 , 0 , 0 }
 };
// size of matrix
 int size = 0;
 for(int i=0; i<4; i++)
 {
    for(int j=0; j<5; j++)
    {
        if(sparse_matrix[i][j]!=0)
        {
            size++;
        }
    }
 }
// Defining final matrix
 int matrix[3][size];
  int k=0;
// Computing final matrix
 for(int i=0; i<4; i++)
 {
    for(int j=0; j<5; j++)
    {
            if(sparse_matrix[i][j]!=0)
            {
                matrix[0][k] = i;
                matrix[1][k] = j;
                matrix[2][k] = sparse_matrix[i][j];
                k++;
            }
      }
    }
   // Displaying the final matrix
    for(int i=0 ;i<3; i++)
    {
        for(int j=0; j<size; j++)
        {
            printf("%d ", matrix[i][j]);
            printf("\t");
        }
        printf("\n");
    }
    return 0;
}
```

OUTPUT



```
0         0         1         1         3         3
2         4         2         3         1         2
7         9         5         7         2         3


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

RESULT AND DISCUSSION

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

VIVA QUESTIONS

1. What is a sparse matrix?
2. How do you represent a sparse matrix?
3. What are the operations performed on sparse matrix?
4. What are the applications of sparse matrix?

EXPERIMENT – 5

SINGLY LINKED LIST

AIM

Write a program to implement Single linked lists.

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,item;

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)
{
case 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");
```

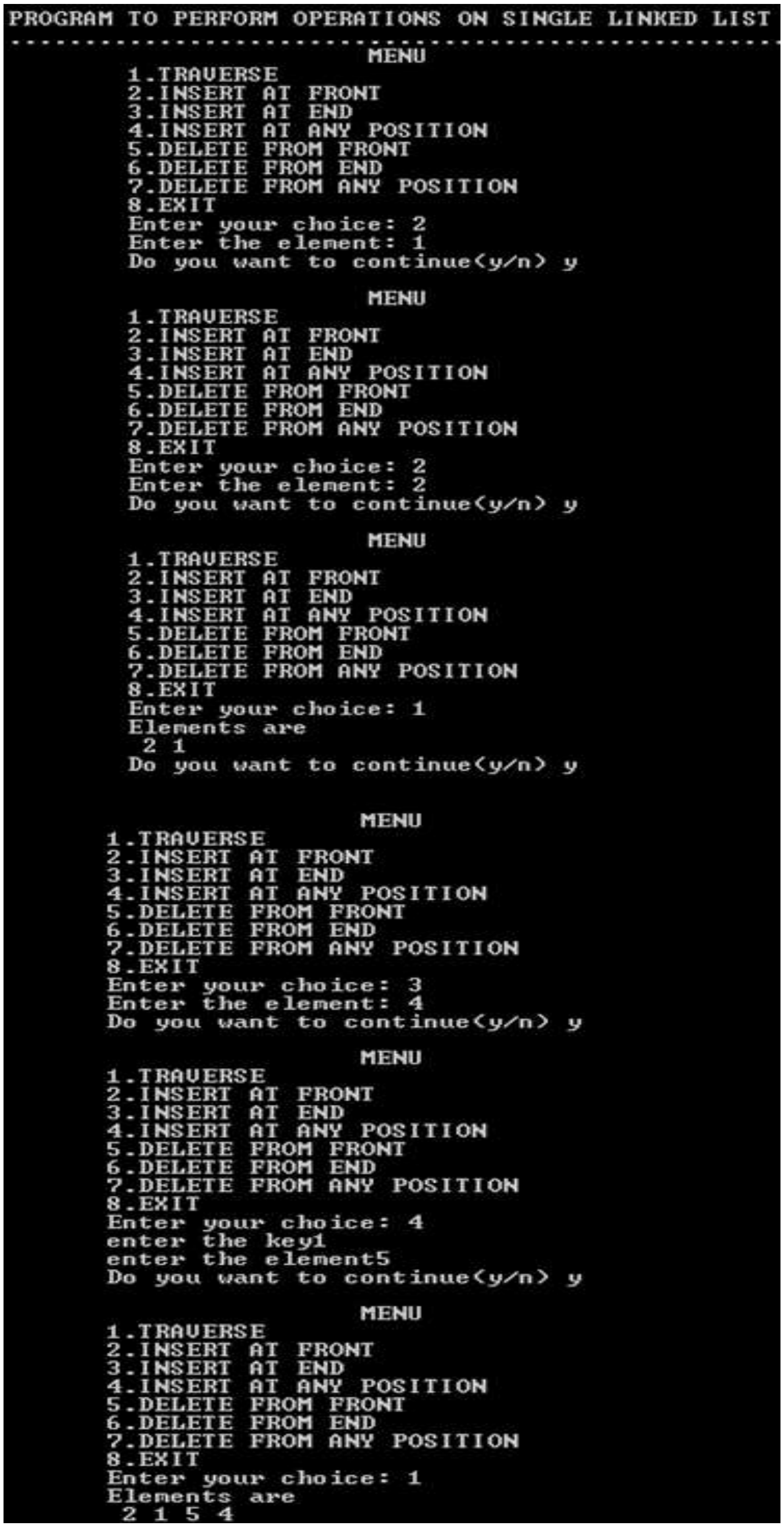
```
        break;
}
printf("\t\tDo you want to continue(y/n) ");
scanf(" %c",&c);
}
while(c=='y'||c=='Y');
getch();
}
void insertfront()
{
newptr=(NODE*)malloc(sizeof(NODE));
printf("\t\tEnter the element: ");
scanf("%d",&newptr->data);
newptr->link=NULL;
if(newptr==NULL)
printf("\t\tInsufficient memmory");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

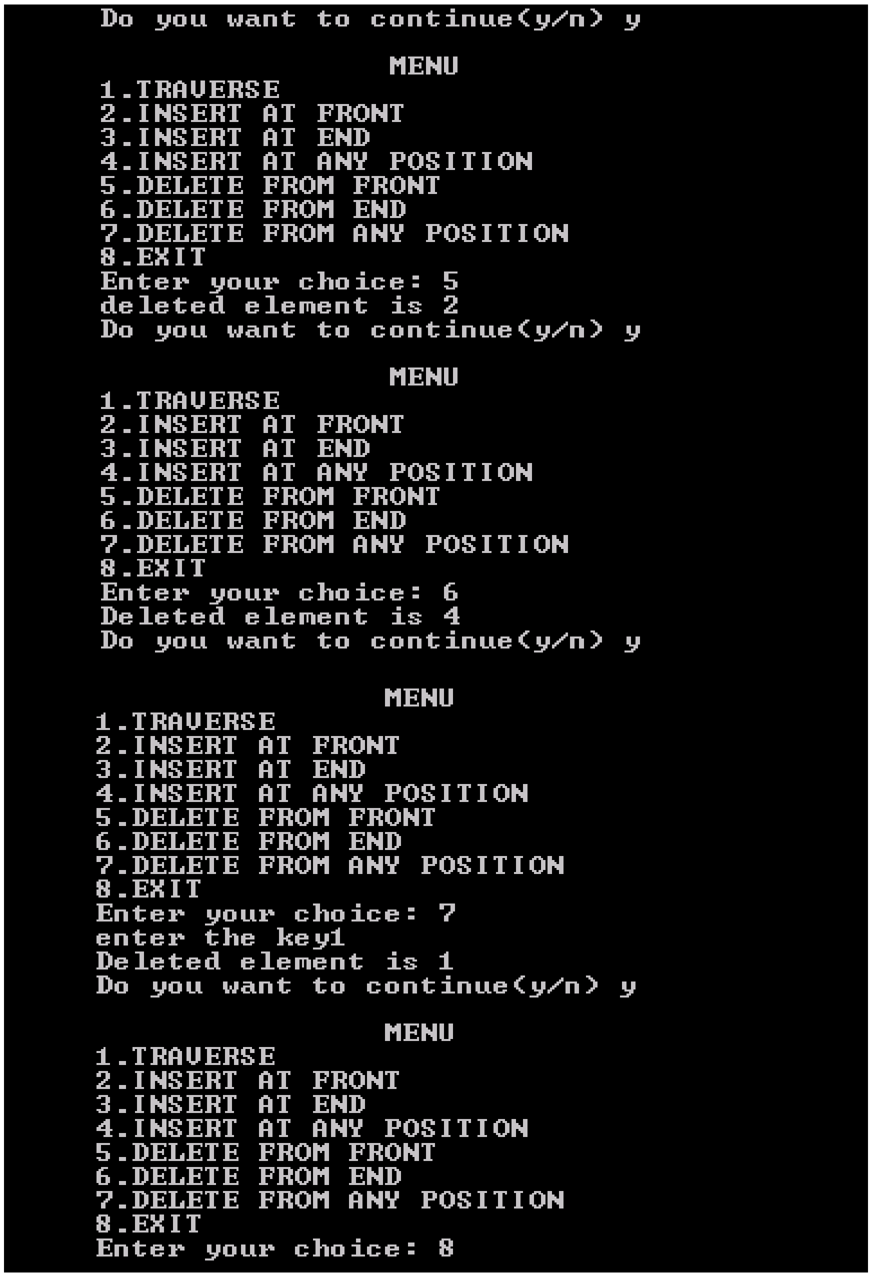
```
{
ptr=header;
ptr1=ptr->link;
if(ptr1==NULL)
printf("\t\tEmpty list");
else
{
while(ptr1->link!=NULL)
{
ptr=ptr->link;
ptr1=ptr1->link;
}
ptr->link=NULL;
printf("\t\tDeleted element is %d",ptr1->data);
free(ptr1);
}
printf("\n");
}
void deleteany()
{
int key;
ptr=header;
ptr1=ptr->link;
if(ptr1==NULL)
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
```

RESULT AND DISCUSSION

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

VIVA QUESTIONS

1. How will you represent a linked list in a graphical view?

2. How many types of Linked List exist?

3. How can you represent a linked list node?

4. Which type of memory allocation is referred for Linked List?

5. What do you know about traversal in linked lists?

6. What are the main differences between the Linked List and Linear Array?

EXPERIMENT – 6

CONVERSION OF EXPRESSIONS

AIM

Write a program to implement infix to postfix expression conversion and its 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

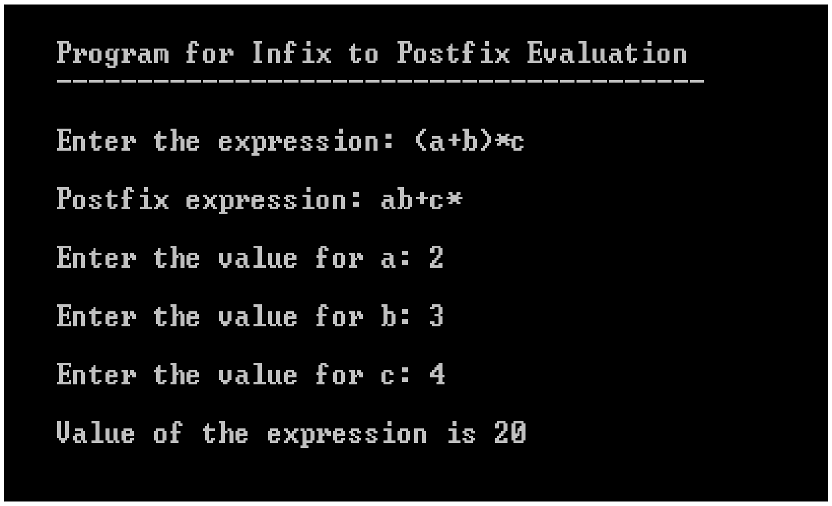
```
#include <stdio.h>
#include<conio.h>
```

```
#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()
{
clrscr();
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[top]);
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]=ch;
}
int priority(char c)
{
if (c=='t'||c=='-')
return 1;
else if(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
```

RESULT AND DISCUSSION

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

VIVA QUESTIONS

1. What is the time complexity of an infix to postfix conversion algorithm?

2. Is Parentheses are simply ignored in the conversion of infix to postfix expression?

3. What are expressions?

4. What do you mean by prefix and postfix expressions?

5. Which data structure is required to check whether an expression contains balanced parenthesis is?

EXPERIMENT – 7

MEMORY ALLOCATION AND GARBAGE COLLECTION USING DOUBLY LINKED LIST

AIM

Write a program to implement memory allocation and garbage collection using doubly linked list.

ALGORITHM

Assume that START is the first element in the linked list and TAIL is the last element of linked list.

i. Insert At Beginning

```
1. Start
2. Input the DATA to be inserted
3. Create a new node.
4. NewNode → Data = DATA NewNode →Lpoint =NULL
5. IF START IS NULL NewNode→ Rpoint = NULL
6. Else NewNode → Rpoint = START START→Lpoint = NewNode
7. START =NewNode
8. Stop
```

ii. Insertion at location:

```
1. Start
2. Input the DATA and POS
3. Initialize TEMP = START; i = 0
4. Repeat the step 4 if (i less than POS) and (TEMP is not equal to NULL)
5. TEMP = TEMP → RPoint; i = i +1
6. If (TEMP not equal to NULL) and (i equal to POS)
```

(a) Create a New Node

(b) NewNode → DATA = DATA

```
(c) NewNode → RPoint = TEMP → RPoint
(d) NewNode → LPoint = TEMP
(e) (TEMP → RPoint) → LPoint = NewNode
    1. (f ) TEMP → RPoint = New Node
    2. Else
(a) Display “Position NOT found”
    1. Stop
```

iii. Insert at End

```
    1. Start
    2. Input DATA to be inserted
    3. Create a NewNode
    4. NewNode → DATA = DATA
    5. NewNode → RPoint = NULL
    6. If (SATRT equal to NULL)
a. START = NewNode
b. NewNode → LPoint=NULL
    1. Else
a. TEMP = START
b. While (TEMP → Next not equal to NULL)
i. TEMP = TEMP → Next
c. TEMP → RPoint = NewNode
d. NewNode → LPoint = TEMP
    1. Stop
```

iv. Forward Traversal

```
1. Start
2. If (START is equal to NULL)
```

```
a) Display "The list is Empty"

b) Stop
```

```
1. Initialize TEMP = START
2. Repeat the step 5 and 6 until (TEMP == NULL )
3. Display "TEMP → DATA"
4. TEMP = TEMP → Next
5. Stop
```

v. Backward Traversal

```
1. Start
2. If (START is equal to NULL)
3. Display "The list is Empty"
4. Stop
5. Initialize TEMP = TAIL
6. Repeat the step 5 and 6 until (TEMP == NULL )
7. Display "TEMP → DATA"
8. TEMP = TEMP → Prev
9. Stop
```

PROGRAM

```
#include <stdio.h>

#include <stdlib.h>

struct node {

  int data;

  struct node *prev;

  struct node *next;

};

struct node *head = NULL;
struct node *last = NULL;
struct node *current = NULL;
//Create Linked List
void insert(int data) {
  // Allocate memory for new node;
  struct node *link = (struct node*) malloc(sizeof(struct node));
  link->data = data;
  link->prev = NULL;
  link->next = NULL;
  // If head is empty, create new list
  if(head==NULL) {
    head = link;
    return;
  }
  current = head;
    // move to the end of the list
  while(current->next!=NULL)
    current = current->next;
  // Insert link at the end of the list
  current->next = link;
  last = link;
  link->prev = current;
}
//display the list
void printList() {
  struct node *ptr = head;


  printf("\n[head] <=>");


  //start from the beginning
  while(ptr->next != NULL) {
    printf(" %d <=>",ptr->data);
    ptr = ptr->next;
  }
  printf(" %d <=>",ptr->data);
  printf(" [head]\n");
}
int main() {
  insert(10);
  insert(20);
  insert(30);
  insert(1);
  insert(40);
  insert(56);
  printList();
  return 0;
}
```

OUTPUT

```
$gcc -o main *.c
$main

[head] <=> 10 <=> 20 <=> 30 <=> 1 <=> 40 <=> 56 <=> [head]
```

RESULT AND DISCUSSION

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

VIVA QUESTIONS

1. What is a linked list?
2. What is the difference between single linked list and double linked list?
3. What are dynamic memory allocations?
4. What do you mean by garbage collection in c?
5. What are the different types of insertions in doubly linked list?

EXPERIMENT – 8

POLYNOMIYAL REPRESENTATION USING LINKED LIST

AIM

Write a program to implement polynomial representation using linked list.

ALGORITHM

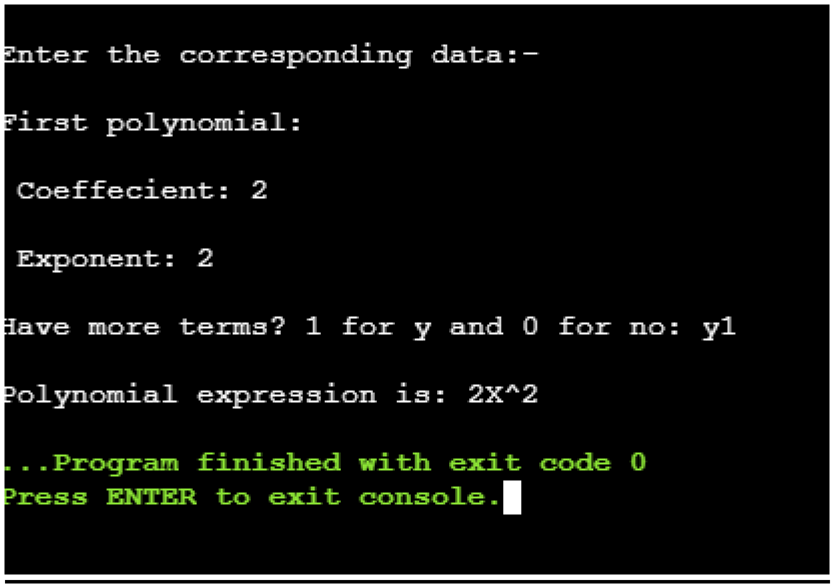
```
Step 1. Start.

Step 2. Define a structure with coeft and degree and a link to next location. And name it as
poly.

Step 3. Declare pointers for root and new nodes.

Step 4. Read the highest degree and its coefficients

Step 6. if(coeft!=0)then do

 set new=(poly*)malloc(sizeof(poly))

            Step 6.1 if(new= =NULL)

                        then print memory allocation error and exit

                                     else

                                    do the following

                                    new->coeft=coeft;

                                                new->degree=hdegree;

                                                new->link=NULL;

Step 7.if(root= =NULL)

 then set

root=new;

temp=root;

else do
```

```
temp->link=new;
temp=new;
Step 8. hdegree--;
Step 9. Print the polynomial
Step 10. Stop.
```

PROGRAM

```
#include<stdio.h>

struct node
{
int coeft;
int degree;
struct node *link;
};
typedef struct node poly;
main()
{
poly *root,*temp,*new;
int hdegree, coeft;
root=NULL;
clrscr();
printf("Enter the highest degree of polynomial:")
scanf("%d",&hdegree);
while(hdegree>=0)
{
printf("Enter coefficient of variable with degree %d",hdegree);
scanf("%d",&coeft);
if(coeft!=0)
{
new=(poly*)malloc(sizeof(poly));
if(new= =NULL)
{
printf("Memory allocation error…"); exit(0);
}
new->coeft=coeft;
new->degree=hdegree;
new->link=NULL;
if(root= =NULL)
{
root=new;
temp=root;
}
else
{
temp->link=new;
temp=new;
}
}
hdegree--;
}
clrscr();
printf("\n The Polynomial is:\n\n");
temp=root;
while(temp!=NULL)
{
if(temp->coeft>0)
 printf("+%dx%d",temp->coeft,temp->degree);
else
  printf("%dx%d",temp->coeft,temp->degree);
temp=temp->link;
}
 getch();
}
```

OUTPUT



```
Enter the corresponding data:-

First polynomial:

 Coeffecient: 2

 Exponent: 2

Have more terms? 1 for y and 0 for no: y1

Polynomial expression is: 2X^2

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

RESULT AND DISCUSSION

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

VIVA QUESTIONS

1. What is a polynomial?

2. What is linked list?

3. How do you represent polynomial using linked list?

4. What are the different types of linked list?

EXPERIMENT – 9
LINEAR SEARCH AND BINARY SEARCH

AIM

Write a program to implement linear search and binary search.

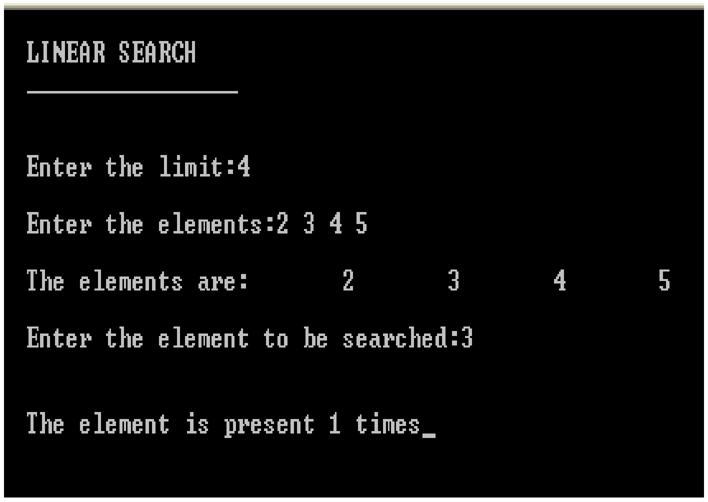
ALGORITHM

```
Step 1: Start
Step 2: Set count as zero, flag as zero and position as zero
Step 3: Read the limit
Step 4: Read the elements
Step 5: Loop (i<n)
    Step 5.1: Read the elements
    Step 5.2: Incriment i
Step 6: End of loop
Step 7: Loop (i<n)
    Step 7.1: Print the elements
Step 8: Loop ends
Step 9: Printf enter the key elements
Step10: Read the element
Step 11: Loop (i<n)
    Step 11.1: If (a[i]+key)
        Step 11.1.1: Set flag =1
        Step 11.1.2: Set position=1+1
        Step 11.1.3: Increment count
        Step 11.1.4: Print the key found at position
    Step 11.2: If ends
Step 12: Loop ends
Step 13: If (flag=1)
    Step 13.1: Print the key found number of times
Step 14: If ends
Step 15: Else
    Step 15.1: Print element not found
Step 16: Else end
Step 17: Stop
```

PROGRAM

```
#include<stdio.h>

#include<conio.h>

void main()

{

 int a[50],i,n,elt,count;

 clrscr();

 printf("\n\tLINEAR 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\tThe elements are:");

for(i=0;i<n;i++)

{

printf("\t%d",a[i]);

}

printf("\n\n\tEnter the element to be searched:");

scanf("%d",&elt);

count=0;

for(i=0;i<n;i++)

{

if(a[i]==elt)
{
count=count+1;
}
}
printf("\n\n\tThe element is present %d times",count);
getch();
}
```

OUTPUT:



```
LINEAR SEARCH
_______________

Enter the limit:4

Enter the elements:2 3 4 5

The elements are:       2       3       4       5

Enter the element to be searched:3


The element is present 1 times_
```

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

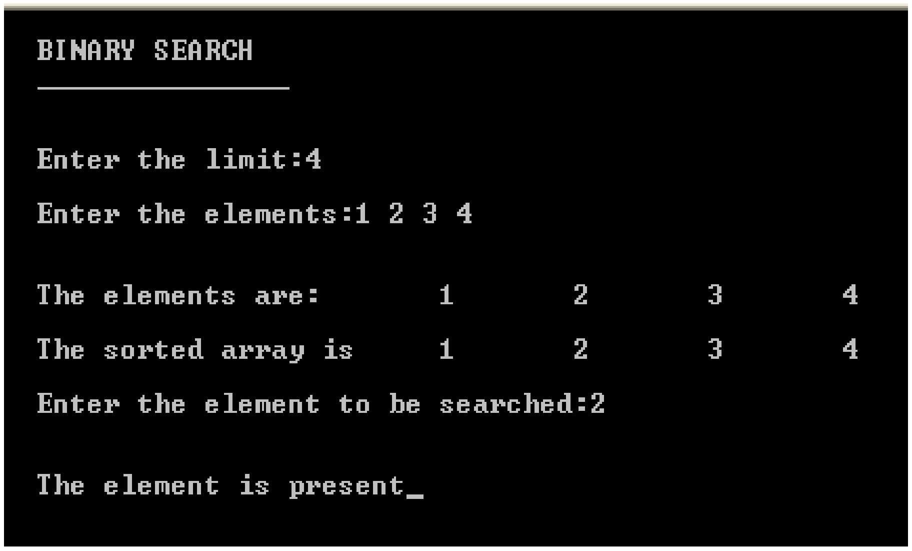
```
#include<stdio.h>

#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");

}

 getch();

}
```

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_
```

RESULT AND DISCUSSION

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

VIVA QUESTIONS

1. What is linear search?

2. What is binary search?

3. What are the complexities of linear and binary search?

4. Which search is better and why?

EXPERIMENT – 10

BINARY TREES

AIM

Write a program to implement binary tree.

ALGORITHM

```
Step 1: Start
Step 2: Ptr=new node ()
Step 3: Repeat the following steps until info!=0
Step 4: Enter the ptr->data
Step 5: Ptr->left=NULL and ptr->right=NULL
Step 6: If (root=NULL)
    Step 6.1: Root =ptr
    Step 6.2: Start=root
Step 7: Else
    Step 7.1: Root=start
    Step 7.2: While (root!=NULL)
        Step 7.2.1: If (ptr->data<root->data)
            Step 7.2.1.1: If (root->left=NULL)
                Step 7.2.1.1.1: Root->left=ptr
                Step 7.2.1.1.2: Exit while loop
        Step 7.2.2: Root=root->left
        Step 7.2.3: Else
            Step 7.2.3.1: If (root->right==NULL)
            Step 7.2.3.2: Root->right=ptr
            Step 7.2.3.3: Exit while loop
            Step 7.2.3.4: Root=root->right
        Step 7.2.4: End if
    Step 7.3: End While
Step 8: End if
Step 9: Stop
```

Preorder ()

```
Step 1: Start
Step 2: If (p=NULL) ie start= NULL
    Step 2.1: Tree traversal not possible
Step 3: Else
        Step 3.1: Print p->data
        Step 3.2: Preorder (p->left)
        Step 3.3: Preorder (p->right)
Step 4: End if
Step 5: Stop
```

Inorder ()

```
Step 1: Start
Step 2: If (p=NULL)
        Step 2.1: Tree traversal not possible
Step 3: Else
        Step 3.1: Inorder (p->left)
        Step 3.2: Print p->data
        Step 3.3: Inorder (p->right)
Step 4: End if
Step 5: Stop
```

Postorder ()

```
Step 1: Start
Step 2: If (p==NULL)
        Step 2.1: Tree traversal not possible
Step 3: Else
        Step 3.1: Postorder (p->left)
        Step 3.2: Postorder (p->right)
        Step 3.3: Print p->data
Step 4: End if
Step 5: Stop
```

PROGRAM

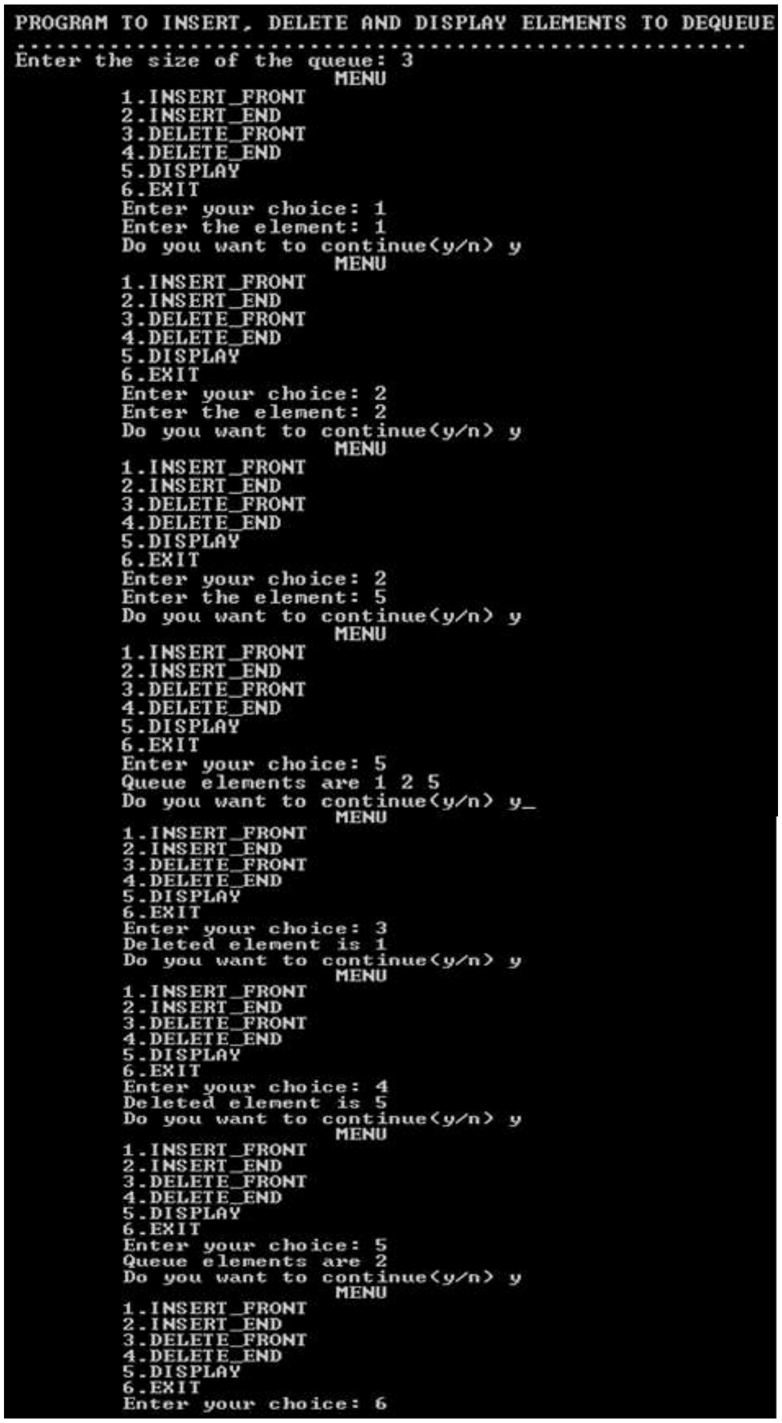
```
# include <stdio.h>

# include <conio.h>

# include <stdlib.h>

 typedef struct BST {
```

```
int data;

struct BST *lchild, *rchild;

} node;

void insert(node *, node *);

void inorder(node *);

void preorder(node *);

void postorder(node *);

node *search(node *, int, node **);

int f=0;

void main() {

int choice;

char ans = 'N';

int key;

node *new_node, *root, *tmp, *parent;

node *get_node();

root = NULL;

clrscr();

printf("\n\n\tProgram For Binary Search Tree");

printf("\n\n\t------------------------------");

do {

printf("\n");

printf("\n\tMenu");

printf("\n\t----");

printf("\n\t1.Create");

printf("\n\t2.Search");

printf("\n\t3.Traversals");
printf("\n\t4.Exit");

printf("\n\tEnter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

do {

new_node = get_node();

printf("\tEnter The Element: ");

scanf("%d", &new_node->data);

if (root == NULL) /* Tree is not Created */

root = new_node;

else

insert(root, new_node);

printf("\tWant To enter More Elements?(y/n): ");

scanf(" %c",ans);

ans = getch();

} while (ans == 'y');

break;

case 2:

printf("\tEnter Element to be searched: ");

scanf("%d", &key);

tmp = search(root, key, &parent);

if(tmp!=NULL)

{

printf("\n\tParent of node %d is %d\n", tmp->data, parent->data);

}
break;
case 3:
 if (root == NULL)
printf("Tree Is Not Created");
else {
printf("\n\tThe Inorder display : ");
inorder(root);
printf("\n\tThe Preorder display : ");
preorder(root);
printf("\n\tThe Postorder display : ");
postorder(root);
}
printf("\n");
break;
}
} while (choice != 4);
/*
 Get new Node
 */
node *get_node() {
node *temp;
temp = (node *) malloc(sizeof(node));
temp->lchild = NULL;
temp->rchild = NULL;
return temp;
/*
```

```
 This function is for creating a binary search tree
 */
void insert(node *root, node *new_node) {
if (new_node->data < root->data) {
if (root->lchild == NULL)
root->lchild = new_node;
else
insert(root->lchild, new_node);
}
if (new_node->data > root->data) {
if (root->rchild == NULL)
root->rchild = new_node;
else
insert(root->rchild, new_node);
}
}
/*
 This function is for searching the node from
 binary Search Tree
 */
node *search(node *root, int key, node **parent) {
node *temp;
temp = root;
while (temp != NULL) {
if (temp->data == key) {
f=1;
printf("\tElement %d is Present", temp->data);
return temp;
}
*parent = temp;
if (temp->data > key)
temp = temp->lchild;
else
temp = temp->rchild;
}
if(f==0)
{
printf("\nElement not [present");
return NULL;
}
}
/*
 This function displays the tree in inorder fashion
 */
void inorder(node *temp) {
if (temp != NULL) {
inorder(temp->lchild);
printf("%d", temp->data);
inorder(temp->rchild);
}
}
/*
```

```
 This function displays the tree in preorder fashion
 */
void preorder(node *temp) {
if (temp != NULL) {
printf("%d", temp->data);
preorder(temp->lchild);
preorder(temp->rchild);
}
}
/*
 This function displays the tree in postorder fashion
 */
void postorder(node *temp) {
if (temp != NULL) {
postorder(temp->lchild);
postorder(temp->rchild);
printf("%d", temp->data);
 }
}
```

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
```

RESULT AND DISCUSSION

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

VIVA QUESTIONS

1. What are non linear data structures?
2. What are the examples of non linear data structures?
3. What are the traversals of trees?
4. What are the different types of trees?
5. What is the difference between tree and a graph?

EXPERIMENT – 11

DFS and BFS

AIM

Write a program to implement DFS and BFS.

ALGORITHM

```
Step 1: Start
Step 2: Enter the data
Step 3: Store the data in an array
Step 4: To find BFS goto step 5
Step 5: Set i=0,j=0
Step 6: Repeat the steps 6 to 6.3 till i<n
        Step 6.1: If vis[i]=0 then vis[i]=I and print data[i]
        Step 6.2: Repeat the steps 6.2 to 6.3 till j<n
        Step 6.3:I f adj[i][j]=1&vis[j]=0,then set vis[j]=1& print data[j]
Step 7: To find DFS goto Step 8
Step 8: Vist[x] = 1
        Step 8.1: Print data[x]
Step 9: Repeat the following steps till j<=n
        Step 9.1: If adj[x][j]=1 and vis[j]=0
        Step 9.2: Dfs (j)
Step 10: Stop
```

PROGRAM

```
#include<stdio.h>

int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];

int delete();
```

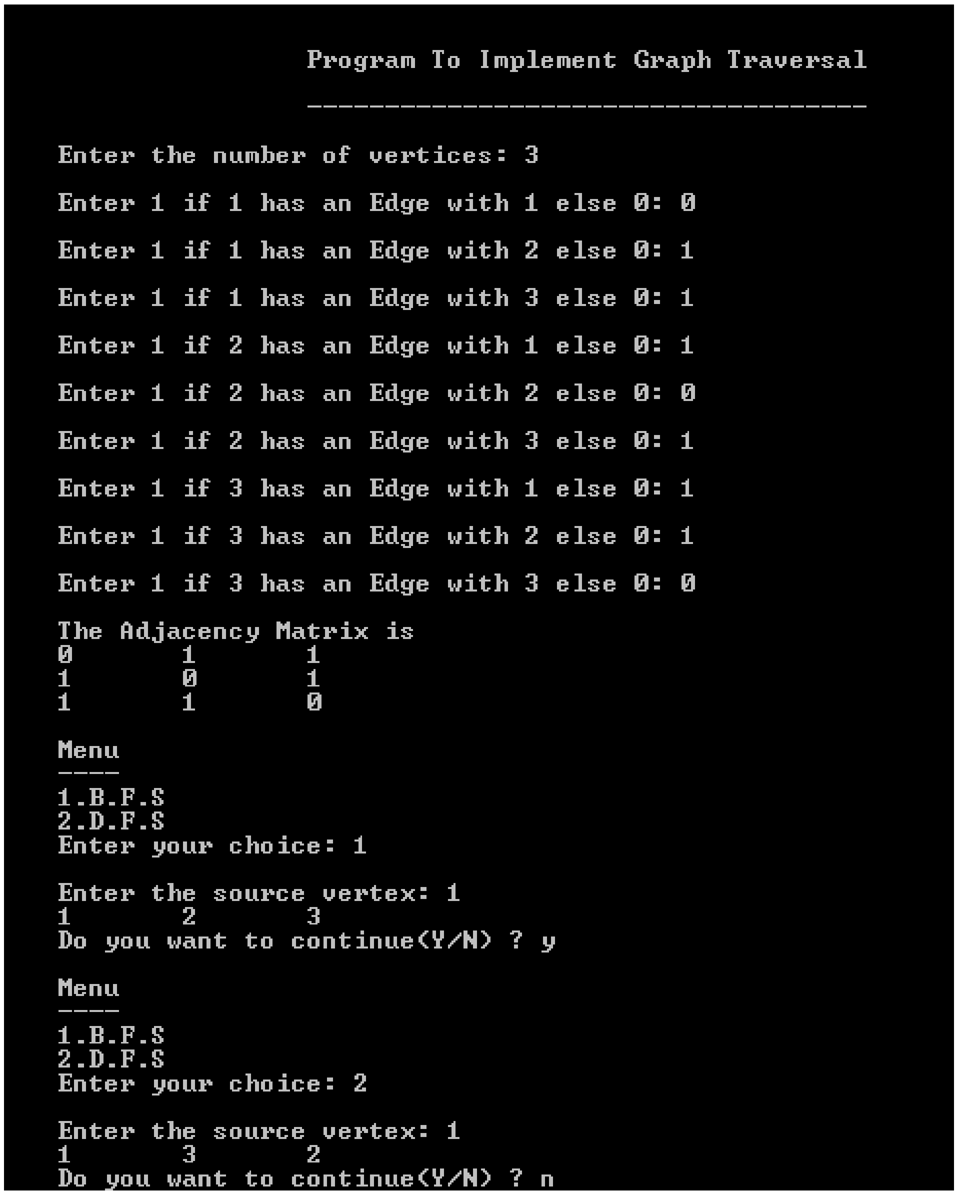
```
void add(int item);

void bfs(int s,int n);

void dfs(int s,int n);

void push(int item);

int pop();


void main()

{

int n,i,s,ch,j;

char c,dummy;

printf("ENTER THE NUMBER VERTICES ");

scanf("%d",&n);

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

printf("ENTER 1 IF %d HAS A NODE WITH %d ELSE 0 ",i,j);

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

}

}

printf("THE ADJACENCY MATRIX IS\n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

printf(" %d",a[i][j]);
}
printf("\n");
}


do
{
for(i=1;i<=n;i++)
vis[i]=0;
printf("\nMENU");
printf("\n1.B.F.S");
printf("\n2.D.F.S");
printf("\nENTER YOUR CHOICE");
scanf("%d",&ch);
printf("ENTER THE SOURCE VERTEX :");
scanf("%d",&s);


switch(ch)
{
case 1:bfs(s,n);
break;
case 2:
dfs(s,n);
break;
}
printf("DO U WANT TO CONTINUE(Y/N) ? ");
scanf("%c",&dummy);
scanf("%c",&c);
}while((c=='y')||(c=='Y'));
}



//**************BFS(breadth-first search) code**************//
void bfs(int s,int n)
{
int p,i;
add(s);
vis[s]=1;
p=delete();
if(p!=0)
printf(" %d",p);
while(p!=0)
{
for(i=1;i<=n;i++)
if((a[p][i]!=0)&&(vis[i]==0))
{
add(i);
vis[i]=1;
}
p=delete();
if(p!=0)
printf(" %d ",p);
}
```

```
for(i=1;i<=n;i++)
if(vis[i]==0)
bfs(i,n);
}
void add(int item)
{
if(rear==19)
printf("QUEUE FULL");
else
{
if(rear==-1)
{
q[++rear]=item;
front++;
}
else
q[++rear]=item;
}
}
int delete()
{
int k;
if((front>rear)||(front==-1))
return(0);
else
{
```

```
k=q[front++];
return(k);
}
}

//***************DFS(depth-first search) code******************//
void dfs(int s,int n)
{
int i,k;
push(s);
vis[s]=1;
k=pop();
if(k!=0)
printf(" %d ",k);
while(k!=0)
{
for(i=1;i<=n;i++)
if((a[k][i]!=0)&&(vis[i]==0))
{
push(i);
vis[i]=1;
}
k=pop();
if(k!=0)
printf(" %d ",k);
}
```

```
for(i=1;i<=n;i++)
if(vis[i]==0)
dfs(i,n);
}
void push(int item)
{
if(top==19)
printf("Stack overflow ");
else
stack[++top]=item;
}
int pop()
{
int k;
if(top==-1)
return(0);
else
{
k=stack[top--];
return(k); } }
```

OUTPUT:



```
                    Program To Implement Graph Traversal
                    ------------------------------------

   Enter the number of vertices: 3

   Enter 1 if 1 has an Edge with 1 else 0: 0

   Enter 1 if 1 has an Edge with 2 else 0: 1

   Enter 1 if 1 has an Edge with 3 else 0: 1

   Enter 1 if 2 has an Edge with 1 else 0: 1

   Enter 1 if 2 has an Edge with 2 else 0: 0

   Enter 1 if 2 has an Edge with 3 else 0: 1

   Enter 1 if 3 has an Edge with 1 else 0: 1

   Enter 1 if 3 has an Edge with 2 else 0: 1

   Enter 1 if 3 has an Edge with 3 else 0: 0

   The Adjacency Matrix is
   0       1       1
   1       0       1
   1       1       0

   Menu
   ----
   1.B.F.S
   2.D.F.S
   Enter your choice: 1

   Enter the source vertex: 1
   1       2       3
   Do you want to continue(Y/N) ? y

   Menu
   ----
   1.B.F.S
   2.D.F.S
   Enter your choice: 2

   Enter the source vertex: 1
   1       3       2
   Do you want to continue(Y/N) ? n
```

RESULT AND DISCUSSION

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

VIVA QUESTIONS

1. What are non linear data structures?

2. What are the examples of non linear data structures?

3. What are the traversals of graphs?

4. What are the differences between DFS and BFS?

5. What is the difference between tree and a graph?

EXPERIMENT – 12

INSERTION SORT

AIM

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
```

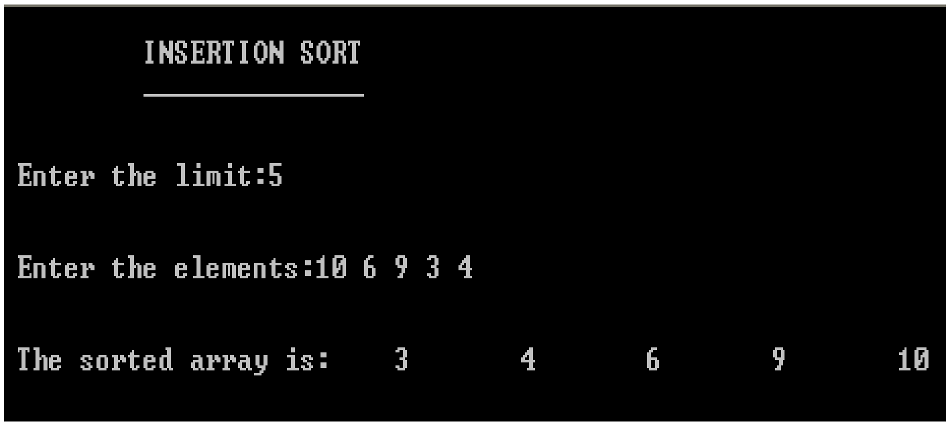
PROGRAM

```
#include<stdio.h>

#include<conio.h>

void main()

{

int a[50],i,n,j,temp;
```

```
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]);
}
```

```
getch();
}
```

OUTPUT:



```
        INSERTION SORT
        ______________

Enter the limit:5

Enter the elements:10 6 9 3 4

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

RESULT AND DISCUSSION

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

VIVA QUESTIONS

1. What is searching?
2. What is the difference between searching and sorting?
3. What are the different types of sorting?
4. Which sorting is considered as better?
5. What is complexity of insertion sort?