DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

DATA STRUCTURES (3S307PC)

LAB MANUAL
2020-2021

II B Tech – I Semester [Branch: CSE]

E.Ravi
Assistant Professor

Pallavi Engineering College
Kuntloor(v), Abdullapurmet(M), Hyderabad, R.R.Dist

Program Outcomes	
PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**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.
PO3	**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.
PO4	**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.
PO5	**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.
PO6	**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.
PO7	**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.
PO8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**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.
PO11	**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.
PO12	**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	
PSO1	**Professional Skills:** The ability to research, understand and implement computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient analysis and design of computer-based systems of varying complexity.
PSO2	**Problem-Solving Skills:** The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.
PSO3	**Successful Career and Entrepreneurship:** The ability to employ modern computer languages, environments, and platforms in creating innovative career paths, to be an entrepreneur, and a zest for higher studies.

DATA STRUCTURES LAB SYLLABUS

.

DATA STRUCTRES

OBJECTIVE:

- It covers various concepts of C programming language
- It introduces searching and sorting algorithms
- It provides an understanding of data structures such as stacks and queues

OUTCOMES:

Upon the completion of Data Structures practical course, the student will be able to:

- Ability to develop C programs for computing and real-life applications using basic elements like control statements, arrays, functions, pointers and strings, and data structures like stacks, queues and linked lists.
- Ability to Implement searching and sorting algorithms

DATA STRUCTURES
LAB

1.Write a program that uses functions to perform the following operations on singly linked list

i) Creation ii) Insertion iii) Deletion iv) Traversal

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

void insertAtBeginning(int);
void insertAtEnd(int);
void insertBetween(int,int,int);
void display();
void removeBeginning();
void removeEnd();
void removeSpecific(int);

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

void main()
{
int choice,value,choice1,loc1,loc2;
clrscr();
while(1){
mainMenu: printf("\n\n****** MENU ******\n1. Insert\n2. Display\n3. Delete\n4. Exit\nEnter your choice: ");
scanf("%d",&choice);
switch(choice)
{
case 1:    printf("Enter the value to be insert: ");
scanf("%d",&value);
while(1){
printf("Where you want to insert: \n1. At Beginning\n2. At End\n3. Between\nEnter your choice: ");
scanf("%d",&choice1);
switch(choice1)
```

```
{
case 1:    insertAtBeginning(value);
break;
case 2:    insertAtEnd(value);
break;
case 3:      printf("Enter the two values where you wanto insert: ");
scanf("%d%d",&loc1,&loc2);
insertBetween(value,loc1,loc2);
break;
default:   printf("\nWrong Input!! Try again!!!\n\n");
goto mainMenu;
}
goto subMenuEnd;
}
subMenuEnd:
break;
case 2:    display();
break;
case 3:    printf("How do you want to Delete: \n1. From Beginning\n2. From End\n3. Spesific\nEnter your choice: ");
scanf("%d",&choice1);
switch(choice1)
{
case 1:    removeBeginning();
break;
case 2:    removeEnd();
break;
case 3:      printf("Enter the value which you wanto delete: ");
scanf("%d",&loc2);
removeSpecific(loc2);
break;
default:   printf("\nWrong Input!! Try again!!!\n\n");
goto mainMenu;
}
break;
case 4:    exit(0);
default: printf("\nWrong input!!! Try again!!\n\n");
}
```

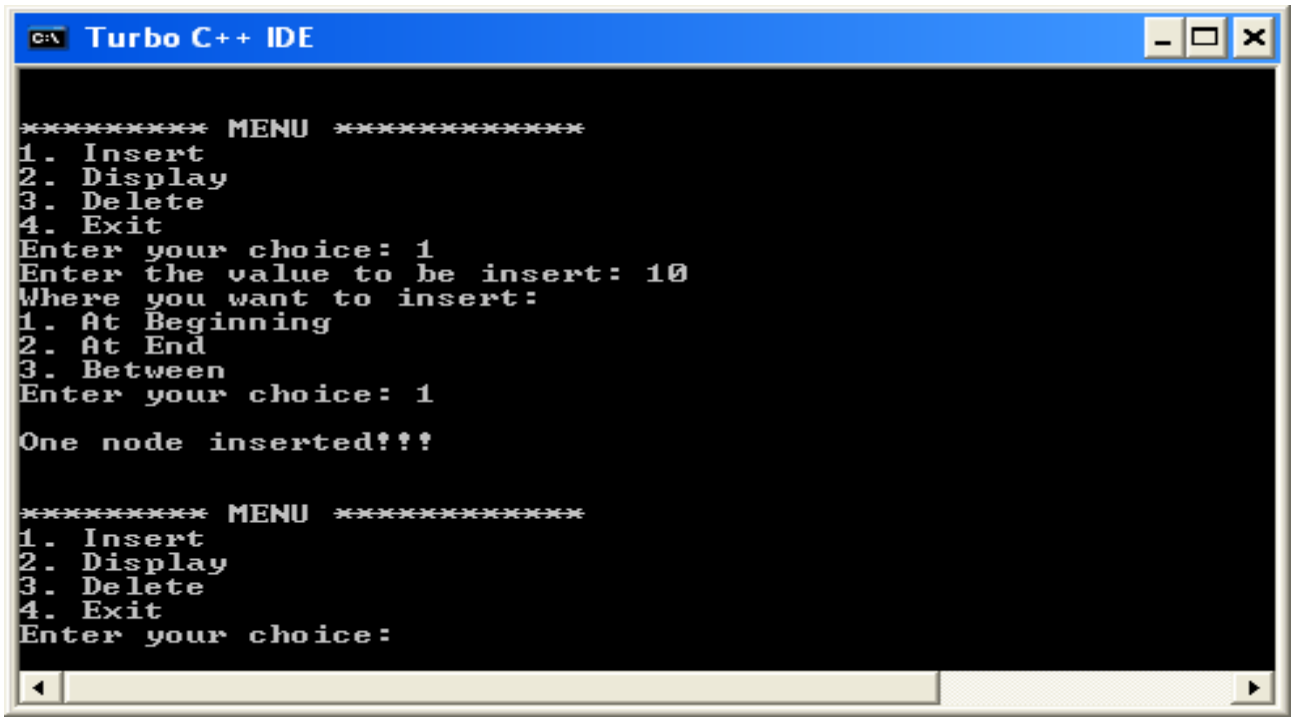
```
}
}

void insertAtBeginning(int value)
{
struct Node *newNode;
newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = value;
if(head == NULL)
{
newNode->next = NULL;
head = newNode;
}
else
{
newNode->next = head;
head = newNode;
}
printf("\nOne node inserted!!!\n");
}
void insertAtEnd(int value)
{
struct Node *newNode;
newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = value;
newNode->next = NULL;
if(head == NULL)
head = newNode;
else
{
struct Node *temp = head;
while(temp->next != NULL)
temp = temp->next;
temp->next = newNode;
}
printf("\nOne node inserted!!!\n");
}
```

```
void insertBetween(int value, int loc1, int loc2)
{
struct Node *newNode;
newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = value;
if(head == NULL)
{
newNode->next = NULL;
head = newNode;
}
else
{
struct Node *temp = head;
while(temp->data != loc1 && temp->data != loc2)
temp = temp->next;
newNode->next = temp->next;
temp->next = newNode;
}
printf("\nOne node inserted!!!\n");
}

void removeBeginning()
{
if(head == NULL)
printf("\n\nList is Empty!!!");
else
{
struct Node *temp = head;
if(head->next == NULL)
{
head = NULL;
free(temp);
}
else
{
head = temp->next;
free(temp);
```

```
printf("\nOne node deleted!!!\n\n");
}
}
}
void removeEnd()
{
if(head == NULL)
{
printf("\nList is Empty!!!\n");
}
else
{
struct Node *temp1 = head,*temp2;
if(head->next == NULL)
head = NULL;
else
{
while(temp1->next != NULL)
{
temp2 = temp1;
temp1 = temp1->next;
}
temp2->next = NULL;
}
free(temp1);
printf("\nOne node deleted!!!\n\n");
}
}
void removeSpecific(int delValue)
{
struct Node *temp1 = head, *temp2;
while(temp1->data != delValue)
{
if(temp1 -> next == NULL){
printf("\nGiven node not found in the list!!!");
goto functionEnd;
}
```

```
temp2 = temp1;
temp1 = temp1 -> next;
}
temp2 -> next = temp1 -> next;
free(temp1);
printf("\nOne node deleted!!!\n\n");
functionEnd:
}
void display()
{
if(head == NULL)
{
printf("\nList is Empty\n");
}
else
{
struct Node *temp = head;
printf("\n\nList elements are - \n");
while(temp->next != NULL)
{
printf("%d --->",temp->data);
temp = temp->next;
}
printf("%d --->NULL",temp->data);
}
}
```

OUT PUT


Turbo C++ IDE
********** MENU **********
1. Insert
2. Display
3. Delete
4. Exit
Enter your choice: 1
Enter the value to be insert: 10
Where you want to insert:
1. At Beginning
2. At End
3. Between
Enter your choice: 1
One node inserted!!!
********** MENU **********
1. Insert
2. Display
3. Delete
4. Exit
Enter your choice:

2.Write a program that uses functions to perform the following operations on doubly linked list.:

i) Creation ii) Insertion iii) Deletion iv) Traversal

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

void insertAtBeginning(int);
void insertAtEnd(int);
void insertAtAfter(int,int);
void deleteBeginning();
void deleteEnd();
void deleteSpecific(int);
void display();

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

void main()
{
  int choice1, choice2, value, location;
  clrscr();
  while(1)
  {
    printf("\n********** MENU **********\n");
    printf("1. Insert\n2. Delete\n3. Display\n4. Exit\nEnter your choice: ");
    scanf("%d",&choice1);
    switch()
    {
      case 1: printf("Enter the value to be inserted: ");
                  scanf("%d",&value);
          while(1)
```

```
        {
                printf("\nSelect from the following Inserting options\n");
                printf("1. At Beginning\n2. At End\n3. After a Node\n4. Cancel\nEnter your choice: ");
          scanf("%d",&choice2);
          switch(choice2)
          {
            case 1:         insertAtBeginning(value);
                            break;
            case 2:         insertAtEnd(value);
                            break;
            case 3:         printf("Enter the location after which you want to insert: ");
                            scanf("%d",&location);
                            insertAfter(value,location);
                            break;
            case 4:         goto EndSwitch;
            default: printf("\nPlease select correct Inserting option!!!\n");
          }
        }
case 2: while(1)
        {
                printf("\nSelect from the following Deleting options\n");
                printf("1. At Beginning\n2. At End\n3. Specific Node\n4. Cancel\nEnter your choice: ");
          scanf("%d",&choice2);
          switch(choice2)
          {
            case 1:         deleteBeginning();
                            break;
            case 2:         deleteEnd();
                            break;
            case 3:         printf("Enter the Node value to be deleted: ");
                            scanf("%d",&location);
                            deleteSpecic(location);
                            break;
            case 4:         goto EndSwitch;
            default: printf("\nPlease select correct Deleting option!!!\n");
```

```
                }
              }
              EndSwitch: break;
        case 3: display();
              break;
        case 4: exit(0);
        default: printf("\nPlease select correct option!!!");
      }
   }
}

void insertAtBeginning(int value)
{
    struct Node *newNode;
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode -> data = value;
    newNode -> previous = NULL;
    if(head == NULL)
    {
      newNode -> next = NULL;
      head = newNode;
    }
    else
    {
      newNode -> next = head;
      head = newNode;
    }
    printf("\nInsertion success!!!");
}
void insertAtEnd(int value)
{
   struct Node *newNode;
   newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode -> data = value;
   newNode -> next = NULL;
```

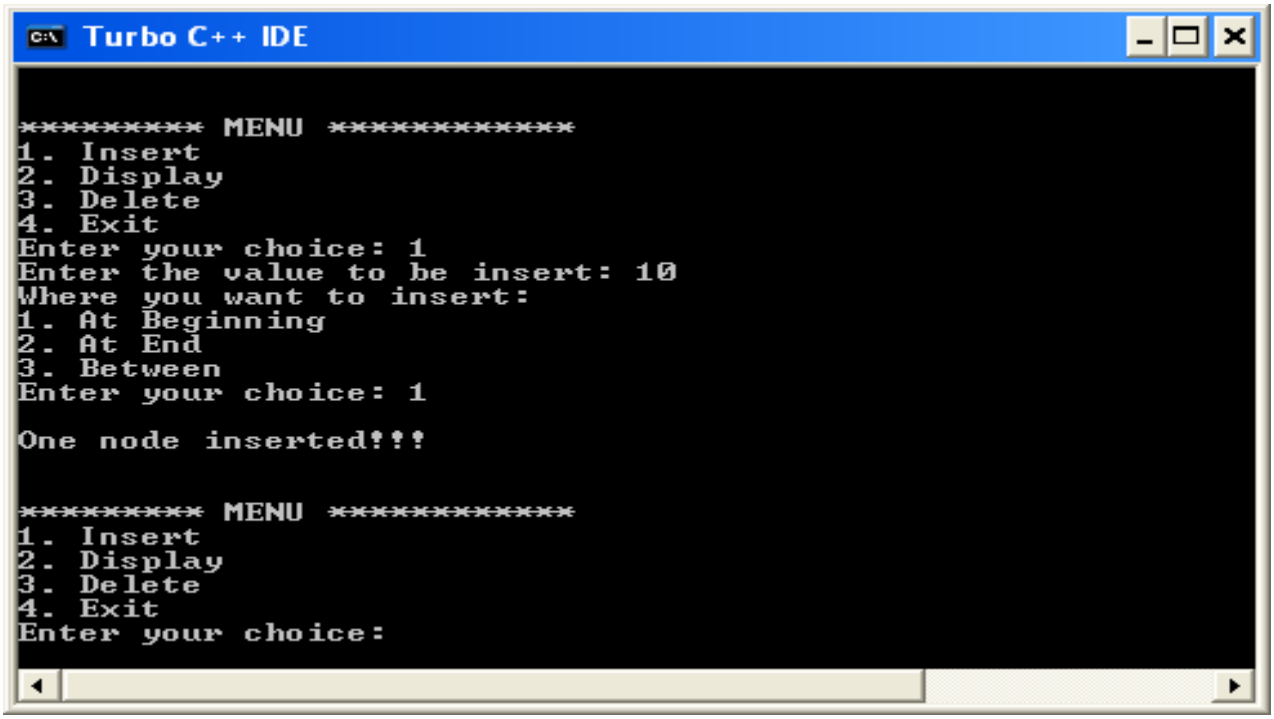
```
  if(head == NULL)
  {
    newNode -> previous = NULL;
    head = newNode;
  }
  else
  {
    struct Node *temp = head;
    while(temp -> next != NULL)
      temp = temp -> next;
    temp -> next = newNode;
    newNode -> previous = temp;
  }
  printf("\nInsertion success!!!");
}
void insertAfter(int value, int location)
{
  struct Node *newNode;
  newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode -> data = value;
  if(head == NULL)
  {
    newNode -> previous = newNode -> next = NULL;
    head = newNode;
  }
  else
  {
    struct Node *temp1 = head, temp2;
    while(temp1 -> data != location)
    {
      if(temp1 -> next == NULL)
      {
        printf("Given node is not found in the list!!!");
        goto EndFunction;
      }
```

```
        else
        {
          temp1 = temp1 -> next;
        }
      }
      temp2 = temp1 -> next;
      temp1 -> next = newNode;
      newNode -> previous = temp1;
      newNode -> next = temp2;
      temp2 -> previous = newNode;
      printf("\nInsertion success!!!");
    }
    EndFunction:
}
void deleteBeginning()
{
    if(head == NULL)
      printf("List is Empty!!! Deletion not possible!!!");
    else
    {
      struct Node *temp = head;
      if(temp -> previous == temp -> next)
      {
        head = NULL;
        free(temp);
      }
      else{
        head = temp -> next;
        head -> previous = NULL;
        free(temp);
      }
      printf("\nDeletion success!!!");
    }
}
void deleteEnd()
```

```
{
  if(head == NULL)
    printf("List is Empty!!! Deletion not possible!!!");
  else
  {
    struct Node *temp = head;
    if(temp -> previous == temp -> next)
    {
      head = NULL;
      free(temp);
    }
    else{
      while(temp -> next != NULL)
        temp = temp -> next;
      temp -> previous -> next = NULL;
      free(temp);
    }
    printf("\nDeletion success!!!");
  }
}
void deleteSpecific(int delValue)
{
  if(head == NULL)
    printf("List is Empty!!! Deletion not possible!!!");
  else
  {
    struct Node *temp = head;
    while(temp -> data != delValue)
    {
      if(temp -> next == NULL)
      {
        printf("\nGiven node is not found in the list!!!");
        goto FuctionEnd;
      }
      else
```

```
        {
          temp = temp -> next;
        }
      }
      if(temp == head)
      {
        head = NULL;
        free(temp);
      }
      else
      {
        temp -> previous -> next = temp -> next;
        free(temp);
      }
      printf("\nDeletion success!!!");
    }
    FuctionEnd:
}
void display()
{
    if(head == NULL)
      printf("\nList is Empty!!!");
    else
    {
      struct Node *temp = head;
      printf("\nList elements are: \n");
      printf("NULL <--- ");
      while(temp -> next != NULL)
      {
        printf("%d <===> ",temp -> data);
      }
      printf("%d ---> NULL", temp -> data);
    }
}
```

OUTPUT


Turbo C++ IDE
********** MENU **********
1. Insert
2. Display
3. Delete
4. Exit
Enter your choice: 1
Enter the value to be insert: 10
Where you want to insert:
1. At Beginning
2. At End
3. Between
Enter your choice: 1
One node inserted!!!
********** MENU **********
1. Insert
2. Display
3. Delete
4. Exit
Enter your choice:

3. Write a program that uses functions to perform the following operations on circular linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal

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

void insertAtBeginning(int);
void insertAtEnd(int);
void insertAtAfter(int,int);
void deleteBeginning();
void deleteEnd();
void deleteSpecific(int);
void display();

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

void main()
{
  int choice1, choice2, value, location;
  clrscr();
  while(1)
  {
    printf("\n********** MENU **********\n");
    printf("1. Insert\n2. Delete\n3. Display\n4. Exit\nEnter your choice: ");
    scanf("%d",&choice1);
    switch()
    {
      case 1: printf("Enter the value to be inserted: ");
                  scanf("%d",&value);
          while(1)
          {
                printf("\nSelect from the following Inserting options\n");
```

```
            printf("1. At Beginning\n2. At End\n3. After a Node\n4. Cancel\nEnter your choice: ");
        scanf("%d",&choice2);
        switch(choice2)
        {
          case 1:       insertAtBeginning(value);
                        break;
          case 2:       insertAtEnd(value);
                        break;
          case 3:       printf("Enter the location after which you want to insert: ");
                        scanf("%d",&location);
                        insertAfter(value,location);
                        break;
          case 4:       goto EndSwitch;
          default: printf("\nPlease select correct Inserting option!!!\n");
        }
      }
case 2: while(1)
      {
            printf("\nSelect from the following Deleting options\n");
            printf("1. At Beginning\n2. At End\n3. Specific Node\n4. Cancel\nEnter your choice: ");
        scanf("%d",&choice2);
        switch(choice2)
        {
          case 1:       deleteBeginning();
                        break;
          case 2:       deleteEnd();
                        break;
          case 3:       printf("Enter the Node value to be deleted: ");
                        scanf("%d",&location);
                        deleteSpecic(location);
                        break;
          case 4:       goto EndSwitch;
          default: printf("\nPlease select correct Deleting option!!!\n");
        }
      }
```

```
                EndSwitch: break;
        case 3: display();
                break;
        case 4: exit(0);
        default: printf("\nPlease select correct option!!!");
      }
   }
}

void insertAtBeginning(int value)
{
    struct Node *newNode;
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode -> data = value;
    if(head == NULL)
    {
      head = newNode;
      newNode -> next = head;
    }
    else
    {
      struct Node *temp = head;
      while(temp -> next != head)
        temp = temp -> next;
      newNode -> next = head;
      head = newNode;
      temp -> next = head;
    }
    printf("\nInsertion success!!!");
}
void insertAtEnd(int value)
{
   struct Node *newNode;
   newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode -> data = value;
```

```
  if(head == NULL)
  {
    head = newNode;
    newNode -> next = head;
  }
  else
  {
    struct Node *temp = head;
    while(temp -> next != head)
      temp = temp -> next;
    temp -> next = newNode;
    newNode -> next = head;
  }
  printf("\nInsertion success!!!");
}
void insertAfter(int value, int location)
{
  struct Node *newNode;
  newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode -> data = value;
  if(head == NULL)
  {
    head = newNode;
    newNode -> next = head;
  }
  else
  {
    struct Node *temp = head;
    while(temp -> data != location)
    {
      if(temp -> next == head)
      {
        printf("Given node is not found in the list!!!");
        goto EndFunction;
      }
```

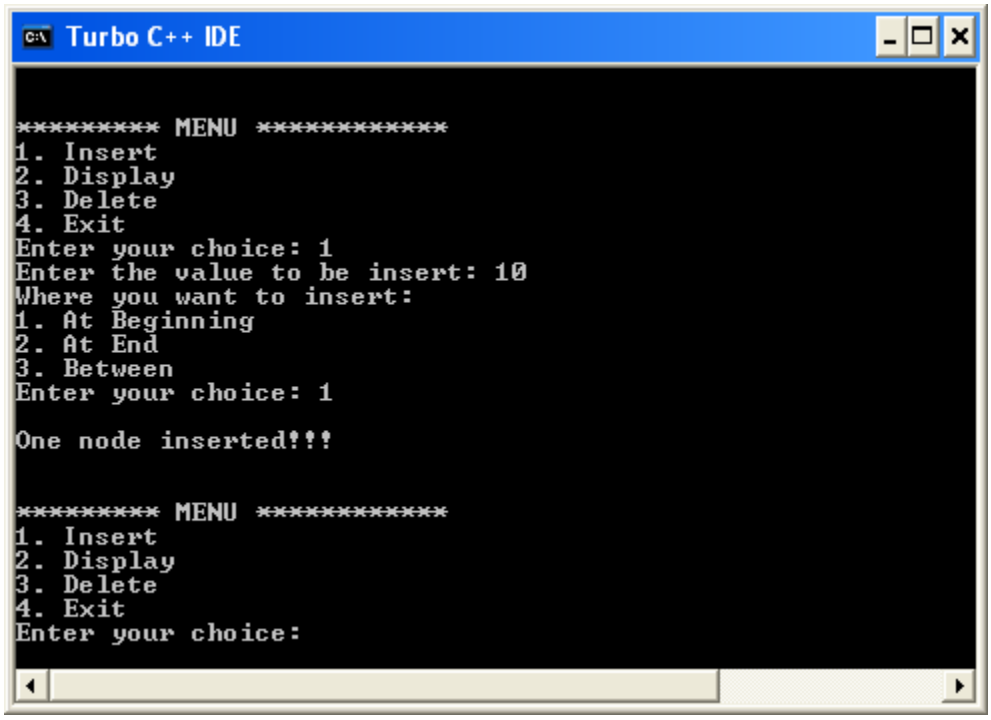
```
        else
        {
          temp = temp -> next;
        }
      }
      newNode -> next = temp -> next;
      temp -> next = newNode;
      printf("\nInsertion success!!!");
    }
    EndFunction:
}
void deleteBeginning()
{
    if(head == NULL)
      printf("List is Empty!!! Deletion not possible!!!");
    else
    {
      struct Node *temp = head;
      if(temp -> next == head)
      {
        head = NULL;
        free(temp);
      }
      else{
        head = head -> next;
        free(temp);
      }
      printf("\nDeletion success!!!");
    }
}
void deleteEnd()
{
    if(head == NULL)
      printf("List is Empty!!! Deletion not possible!!!");
    else
```

```
  {
    struct Node *temp1 = head, temp2;
    if(temp1 -> next == head)
    {
      head = NULL;
      free(temp1);
    }
    else{
      while(temp1 -> next != head){
        temp2 = temp1;
        temp1 = temp1 -> next;
      }
      temp2 -> next = head;
      free(temp1);
    }
    printf("\nDeletion success!!!");
  }
}
void deleteSpecific(int delValue)
{
  if(head == NULL)
    printf("List is Empty!!! Deletion not possible!!!");
  else
  {
    struct Node *temp1 = head, temp2;
    while(temp1 -> data != delValue)
    {
      if(temp1 -> next == head)
      {
        printf("\nGiven node is not found in the list!!!");
        goto FuctionEnd;
      }
      else
      {
        temp2 = temp1;
```

```
        temp1 = temp1 -> next;
      }
    }
    if(temp1 -> next == head){
      head = NULL;
      free(temp1);
    }
    else{
      if(temp1 == head)
      {
        temp2 = head;
        while(temp2 -> next != head)
          temp2 = temp2 -> next;
        head = head -> next;
        temp2 -> next = head;
        free(temp1);
      }
      else
      {
        if(temp1 -> next == head)
        {
          temp2 -> next = head;
        }
        else
        {
          temp2 -> next = temp1 -> next;
        }
        free(temp1);
      }
    }
    printf("\nDeletion success!!!");
  }
  FuctionEnd:
}
void display()
```

```
{
  if(head == NULL)
    printf("\nList is Empty!!!");
  else
  {
    struct Node *temp = head;
    printf("\nList elements are: \n");
    while(temp -> next != head)
    {
      printf("%d ---> ",temp -> data);
    }
    printf("%d ---> %d", temp -> data, head -> data);
  }
}
```

OUTPUT



4.Write a program that implement stack (its operations) using
i) Arrays ii) Pointers

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

void insertAtBeginning(int);
void insertAtEnd(int);
void insertBetween(int,int,int);
void display();
void removeBeginning();
void removeEnd();
void removeSpecific(int);

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

void main()
{
  int choice,value,choice1,loc1,loc2;
  clrscr();
  while(1){
  mainMenu: printf("\n\n****** MENU ******\n1. Insert\n2. Display\n3. Delete\n4. Exit\nEnter your choice: ");
  scanf("%d",&choice);
  switch(choice)
  {
    case 1:          printf("Enter the value to be insert: ");
                     scanf("%d",&value);
                     while(1){
                     printf("Where you want to insert: \n1. At Beginning\n2. At End\n3. Between\nEnter your
choice: ");
                     scanf("%d",&choice1);
```

```
                switch(choice1)
                {
                  case 1: insertAtBeginning(value);
                                  break;
                  case 2: insertAtEnd(value);
                                  break;
                  case 3:     printf("Enter the two values where you wanto insert: ");
                                  scanf("%d%d",&loc1,&loc2);
                                  insertBetween(value,loc1,loc2);
                                  break;
                  default:        printf("\nWrong Input!! Try again!!!\n\n");
                                  goto mainMenu;
                }
                goto subMenuEnd;
                }
                subMenuEnd:
                break;
    case 2:     display();
                break;
    case 3:     printf("How do you want to Delete: \n1. From Beginning\n2. From End\n3. Spesific\nEnter your
choice: ");
                scanf("%d",&choice1);
                switch(choice1)
                {
                  case 1: removeBeginning();
                                  break;
                  case 2: removeEnd();
                                  break;
                  case 3:     printf("Enter the value which you wanto delete: ");
                                  scanf("%d",&loc2);
                                  removeSpecific(loc2);
                                  break;
                  default:        printf("\nWrong Input!! Try again!!!\n\n");
                                  goto mainMenu;
                }
```

```
                        break;
    case 4:             exit(0);
    default: printf("\nWrong input!!! Try again!!\n\n");
   }
   }
}

void insertAtBeginning(int value)
{
   struct Node *newNode;
   newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   if(head == NULL)
   {
     newNode->next = NULL;
     head = newNode;
   }
   else
   {
     newNode->next = head;
     head = newNode;
   }
   printf("\nOne node inserted!!!\n");
}
void insertAtEnd(int value)
{
   struct Node *newNode;
   newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->next = NULL;
   if(head == NULL)
            head = newNode;
   else
   {
     struct Node *temp = head;
```

```
    while(temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
  }
  printf("\nOne node inserted!!!\n");
}
void insertBetween(int value, int loc1, int loc2)
{
  struct Node *newNode;
  newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = value;
  if(head == NULL)
  {
    newNode->next = NULL;
    head = newNode;
  }
  else
  {
    struct Node *temp = head;
    while(temp->data != loc1 && temp->data != loc2)
        temp = temp->next;
    newNode->next = temp->next;
    temp->next = newNode;
  }
  printf("\nOne node inserted!!!\n");
}

void removeBeginning()
{
  if(head == NULL)
        printf("\n\nList is Empty!!!");
  else
  {
    struct Node *temp = head;
    if(head->next == NULL)
```

```
    {
            head = NULL;
            free(temp);
    }
    else
    {
            head = temp->next;
            free(temp);
            printf("\nOne node deleted!!!\n\n");
    }
  }
}
void removeEnd()
{
  if(head == NULL)
  {
    printf("\nList is Empty!!!\n");
  }
  else
  {
    struct Node *temp1 = head,*temp2;
    if(head->next == NULL)
            head = NULL;
    else
    {
            while(temp1->next != NULL)
            {
              temp2 = temp1;
              temp1 = temp1->next;
            }
            temp2->next = NULL;
    }
    free(temp1);
    printf("\nOne node deleted!!!\n\n");
  }
```

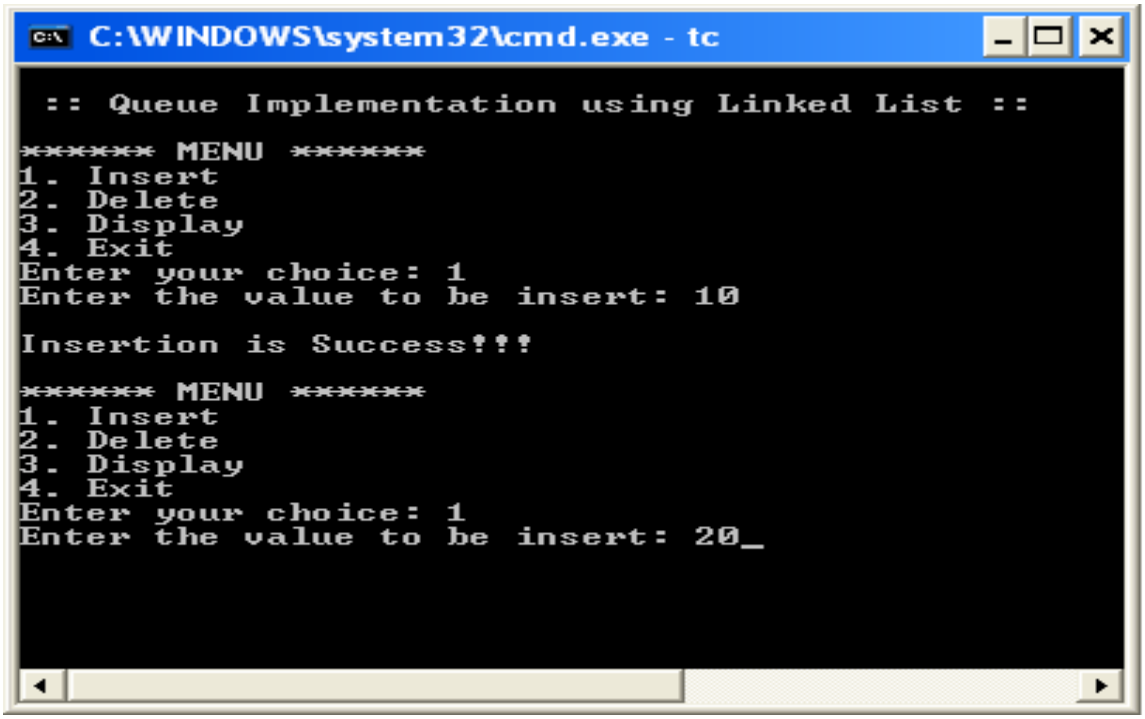
```
}
void removeSpecific(int delValue)
{
  struct Node *temp1 = head, *temp2;
  while(temp1->data != delValue)
  {
    if(temp1 -> next == NULL){
          printf("\nGiven node not found in the list!!!");
          goto functionEnd;
    }
    temp2 = temp1;
    temp1 = temp1 -> next;
  }
  temp2 -> next = temp1 -> next;
  free(temp1);
  printf("\nOne node deleted!!!\n\n");
  functionEnd:
}
void display()
{
  if(head == NULL)
  {
    printf("\nList is Empty\n");
  }
  else
  {
    struct Node *temp = head;
    printf("\n\nList elements are - \n");
    while(temp->next != NULL)
    {
          printf("%d --->",temp->data);
          temp = temp->next;
    }
    printf("%d --->NULL",temp->data);
  }}
```

5. Write a program that implement Queue (its operations) using
   i) Arrays ii) Pointers

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

struct Node
{
  int data;
  struct Node *next;
}*front = NULL,*rear = NULL;

void insert(int);
void delete();
void display();

void main()
{
  int choice, value;
  clrscr();
  printf("\n:: Queue Implementation using Linked List ::\n");
  while(1){
    printf("\n****** MENU ******\n");
    printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");
    printf("Enter your choice: ");
    scanf("%d",&choice);
    switch(choice){
        case 1: printf("Enter the value to be insert: ");
                scanf("%d", &value);
                insert(value);
                break;
        case 2: delete(); break;
        case 3: display(); break;
        case 4: exit(0);
        default: printf("\nWrong selection!!! Please try again!!!\n");
    }
```

```
    }
}
void insert(int value)
{
    struct Node *newNode;
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode -> next = NULL;
    if(front == NULL)
        front = rear = newNode;
    else{
        rear -> next = newNode;
        rear = newNode;
    }
    printf("\nInsertion is Success!!!\n");
}
void delete()
{
    if(front == NULL)
        printf("\nQueue is Empty!!!\n");
    else{
        struct Node *temp = front;
        front = front -> next;
        printf("\nDeleted element: %d\n", temp->data);
        free(temp);
    }
}
void display()
{
    if(front == NULL)
        printf("\nQueue is Empty!!!\n");
    else{
        struct Node *temp = front;
        while(temp->next != NULL){
                printf("%d--->",temp->data);
```

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

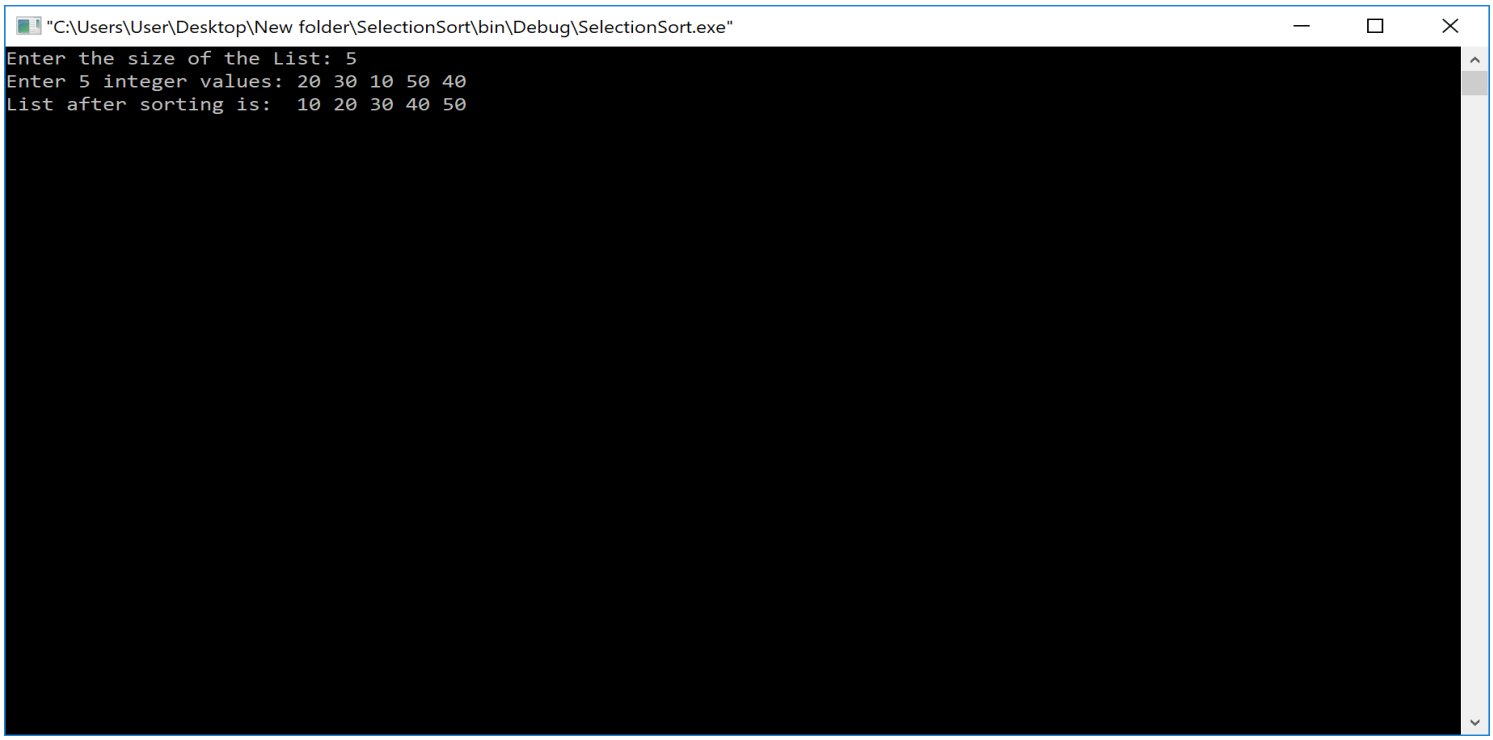


6. Write a program that implements the following sorting methods to sort a given list of integers in ascending order

i) Bubble sort ii) Selection sort iii) Insertion sort

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

void main(){

  int size,i,j,temp,list[100];
  clrscr();

  printf("Enter the size of the List: ");
  scanf("%d",&size);

  printf("Enter %d integer values: ",size);
  for(i=0; i<size; i++)
     scanf("%d",&list[i]);

  //Selection sort logic

  for(i=0; i<size; i++){
     for(j=i+1; j<size; j++){
        if(list[i] > list[j])
              {
           temp=list[i];
           list[i]=list[j];
           list[j]=temp;
        }
     }
  }

  printf("List after sorting is: ");
  for(i=0; i<size; i++)
     printf(" %d",list[i]);
```

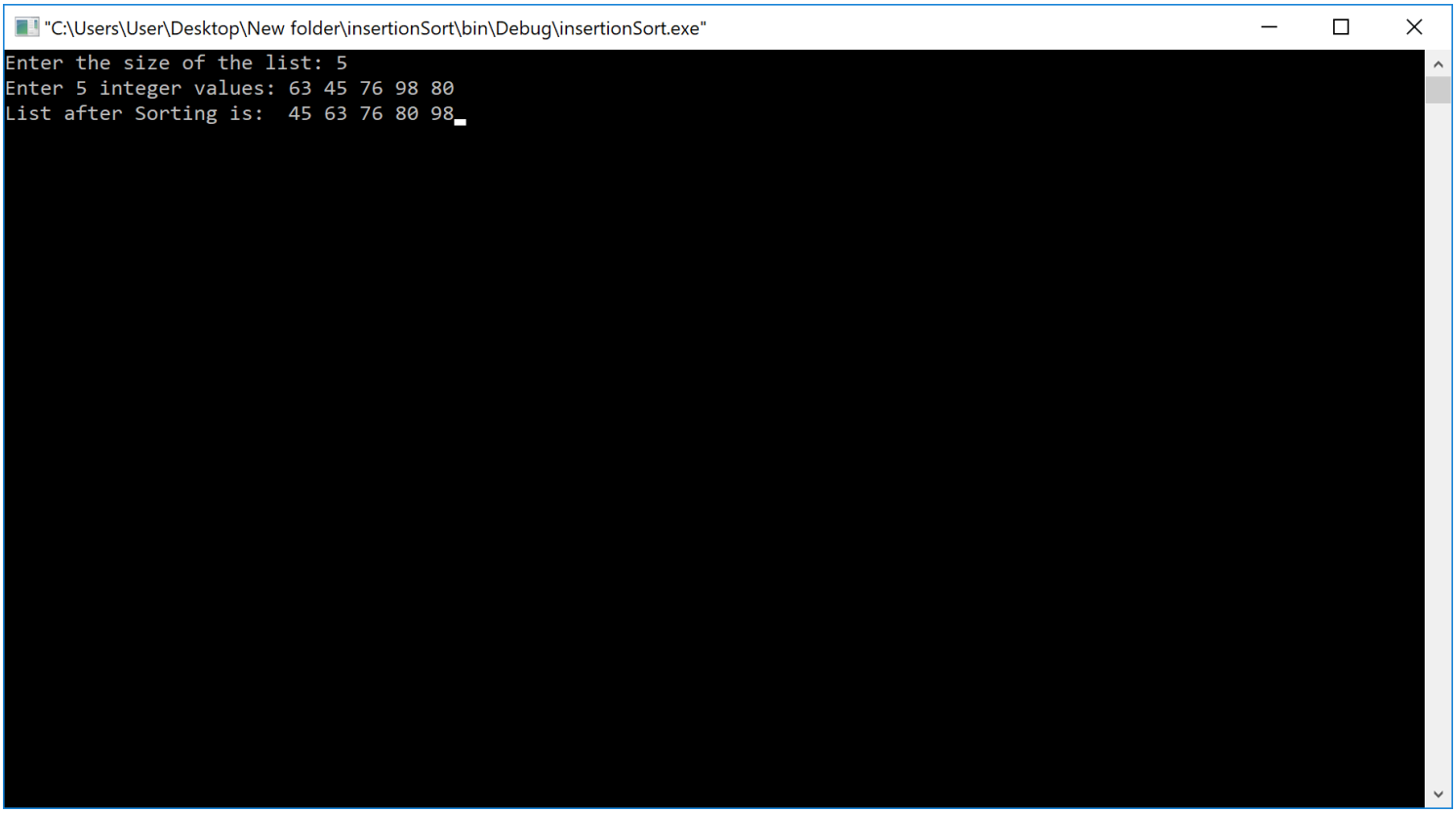
```
getch();
}
```



ii) Insertion Sort

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

void main(){

   int size, i, j, temp, list[100];

   printf("Enter the size of the list: ");
   scanf("%d", &size);

   printf("Enter %d integer values: ", size);
   for (i = 0; i < size; i++)
      scanf("%d", &list[i]);
```

```
  //Insertion sort logic
  for (i = 1; i < size; i++) {
    temp = list[i];
    j = i - 1;
    while ((temp < list[j]) && (j >= 0)) {
      list[j + 1] = list[j];
      j = j - 1;
    }
    list[j + 1] = temp;
  }

  printf("List after Sorting is: ");
  for (i = 0; i < size; i++)
    printf(" %d", list[i]);

 getch();
}
```

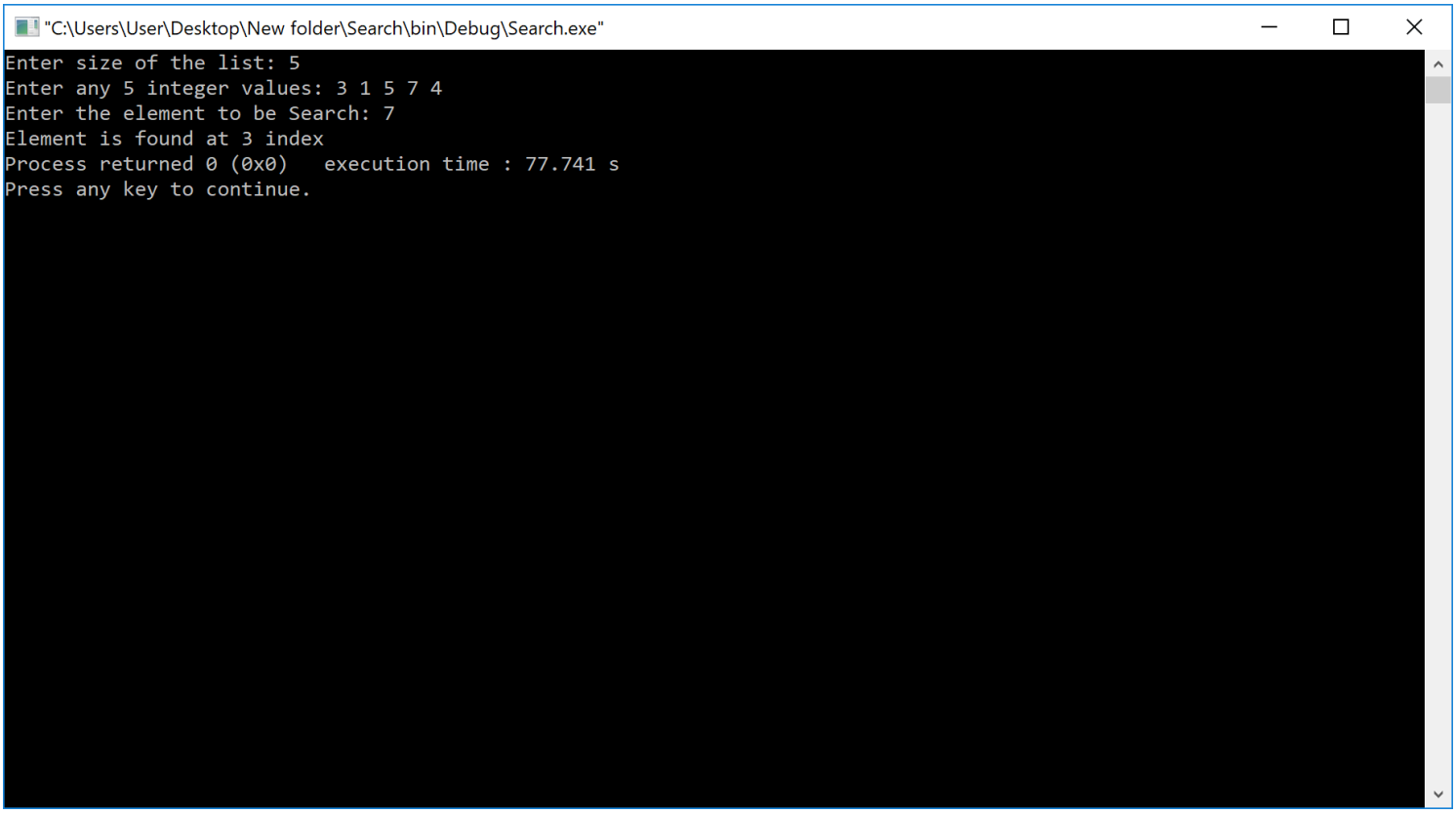


7. Write a program that use both recursive and non recursive functions to perform the following searching operations for a Key value in a given list of integers:

i) Linear search ii) Binary search

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

void main(){
  int list[20],size,i,sElement;

  printf("Enter size of the list: ");
  scanf("%d",&size);

  printf("Enter any %d integer values: ",size);
  for(i = 0; i < size; i++)
    scanf("%d",&list[i]);

  printf("Enter the element to be Search: ");
  scanf("%d",&sElement);
```

```
  // Linear Search Logic
  for(i = 0; i < size; i++)
  {
     if(sElement == list[i])
     {
        printf("Element is found at %d index", i);
        break;
     }
  }
  if(i == size)
     printf("Given element is not found in the list!!!");
  getch();
}
```

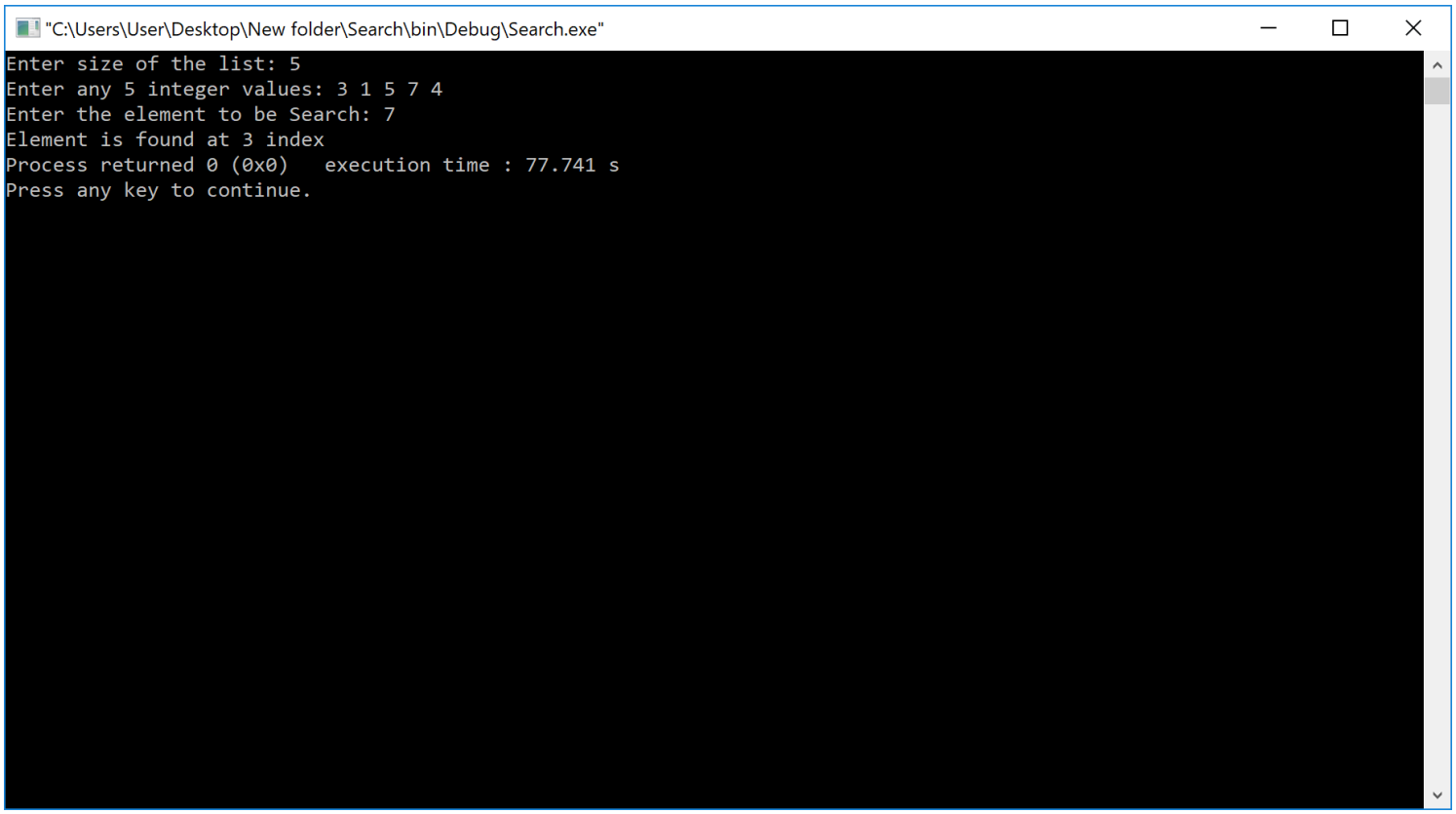


ii) Binary Search

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

```
void main()
{
  int first, last, middle, size, i, sElement, list[100];
  clrscr();

  printf("Enter the size of the list: ");
  scanf("%d",&size);

  printf("Enter %d integer values in Assending order\n", size);

  for (i = 0; i < size; i++)
    scanf("%d",&list[i]);

  printf("Enter value to be search: ");
  scanf("%d", &sElement);

  first = 0;
  last = size - 1;
  middle = (first+last)/2;

  while (first <= last) {
    if (list[middle] < sElement)
      first = middle + 1;
    else if (list[middle] == sElement) {
      printf("Element found at index %d.\n",middle);
      break;
    }
    else
      last = middle - 1;

    middle = (first + last)/2;
  }
  if (first > last)
    printf("Element Not found in the list.");
  getch();  }
```

OUTPUT

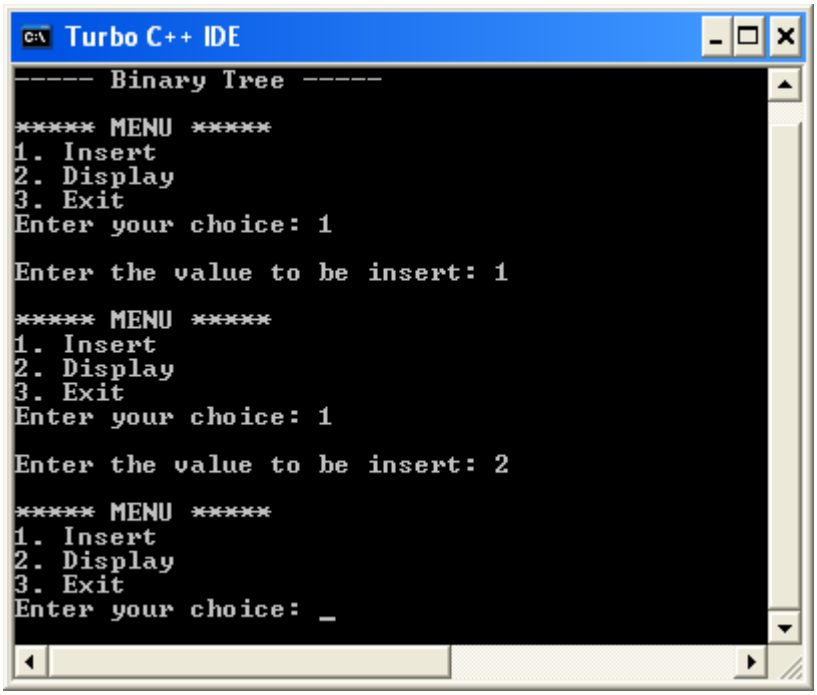


8.Write a program to implement the tree traversal methods.

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

struct Node{
   int data;
   struct Node *left;
   struct Node *right;
};

struct Node *root = NULL;
int count = 0;

struct Node* insert(struct Node*, int);
void display(struct Node*);
```

```
void main(){
  int choice, value;
  clrscr();
  printf("\n----- Binary Tree -----\n");
  while(1){
    printf("\n***** MENU *****\n");
    printf("1. Insert\n2. Display\n3. Exit");
    printf("\nEnter your choice: ");
    scanf("%d",&choice);
    switch(choice){
        case 1: printf("\nEnter the value to be insert: ");
                scanf("%d", &value);
                root = insert(root,value);
                break;
        case 2: display(root); break;
        case 3: exit(0);
        default: printf("\nPlease select correct operations!!!\n");
    }
  }
}

struct Node* insert(struct Node *root,int value){
  struct Node *newNode;
  newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = value;
  if(root == NULL){
    newNode->left = newNode->right = NULL;
    root = newNode;
    count++;
  }
  else{
    if(count%2 != 0)
        root->left = insert(root->left,value);
    else
        root->right = insert(root->right,value);
```

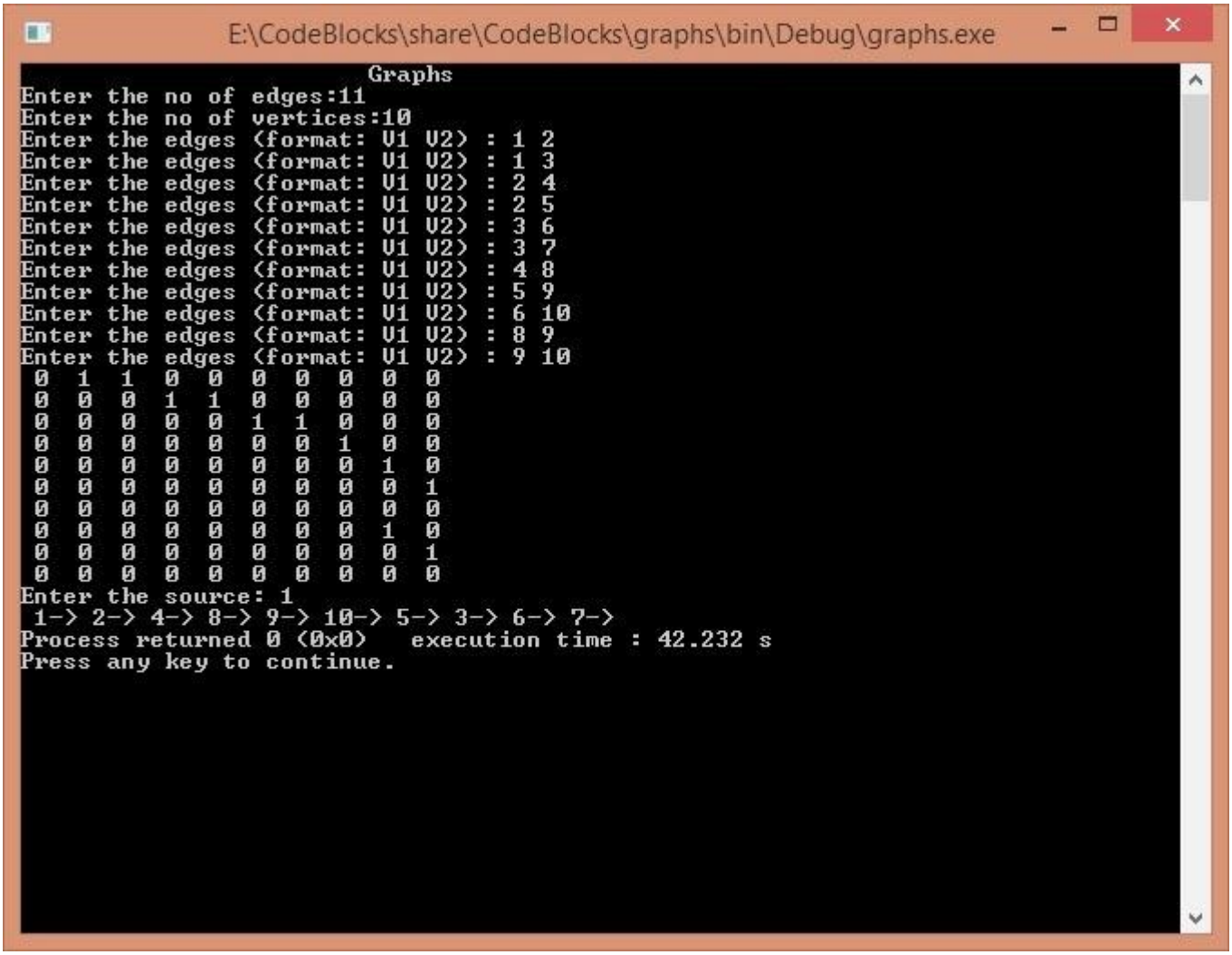
```
  }
  return root;
}
// display is performed by using Inorder Traversal
void display(struct Node *root)
{
  if(root != NULL){
    display(root->left);
    printf("%d\t",root->data);
    display(root->right);
  }
}
```



9. Write a program to implement the graph traversal methods.

```
#include <stdio.h>
#include <stdlib.h>
int source,V,E,time,visited[20],G[20][20];
void DFS(int i)
{
   int j;
   visited[i]=1;
   printf(" %d->",i+1);
   for(j=0;j<V;j++)
   {
      if(G[i][j]==1&&visited[j]==0)
         DFS(j);
   }
}
int main()
{
   int i,j,v1,v2;
   printf("\t\t\tGraphs\n");
   printf("Enter the no of edges:");
   scanf("%d",&E);
   printf("Enter the no of vertices:");
   scanf("%d",&V);
   for(i=0;i<V;i++)
   {
      for(j=0;j<V;j++)
         G[i][j]=0;
   }
   for(i=0;i<E;i++)
   {
      printf("Enter the edges (format: V1 V2) : ");
      scanf("%d%d",&v1,&v2);
      G[v1-1][v2-1]=1;

   }

   for(i=0;i<V;i++)
   {
      for(j=0;j<V;j++)
         printf(" %d ",G[i][j]);
      printf("\n");
   }
   printf("Enter the source: ");
   scanf("%d",&source);
      DFS(source-1);
   return 0;
}
```

OUTPUT


E:\CodeBlocks\share\CodeBlocks\graphs\bin\Debug\graphs.exe
Graphs
Enter the no of edges:11
Enter the no of vertices:10
Enter the edges (format: V1 V2) : 1 2
Enter the edges (format: V1 V2) : 1 3
Enter the edges (format: V1 V2) : 2 4
Enter the edges (format: V1 V2) : 2 5
Enter the edges (format: V1 V2) : 3 6
Enter the edges (format: V1 V2) : 3 7
Enter the edges (format: V1 V2) : 4 8
Enter the edges (format: V1 V2) : 5 9
Enter the edges (format: V1 V2) : 6 10
Enter the edges (format: V1 V2) : 8 9
Enter the edges (format: V1 V2) : 9 10
0 1 1 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0
Enter the source: 1
1-> 2-> 4-> 8-> 9-> 10-> 5-> 3-> 6-> 7->
Process returned 0 (0x0) execution time : 42.232 s
Press any key to continue.

****