LOVELY PROFESSIONAL UNIVERSITY

LAB EVALUATION-3 ON DATA STRUCTURE COURSE CODE: CAP-282

SUBMITTED TO

Balraj Kumar
HOD, Dept. of Computer Application
Faculty of Research & Development
Lovely Professional University

SUBMITTED BY

Amlan Kumar Sarkar
Registration No:11900435
Section: D1904
Roll No: RD1904A01
Lovely Professional University

Date of Submission: 04-15-2020

Set-ODD

1. Develop a menu driven program to implement **Binary Search Tree** using linked representation and perform the following operations on it:
   - Insertion Operation
   - Preorder traversal
   - Inorder traversal
   - Postorder traversal
2. Develop a menu driven program to implement **Linear Queue** using linked representation and perform the following operations on it:
   - Enqueue Operation
   - Dequeue Operation
   - Traversal Operation
3. Develop a program to implement **Insertion Sort** to sort an array of elements in descending order.

Answer

Solution 1:

Source Code:

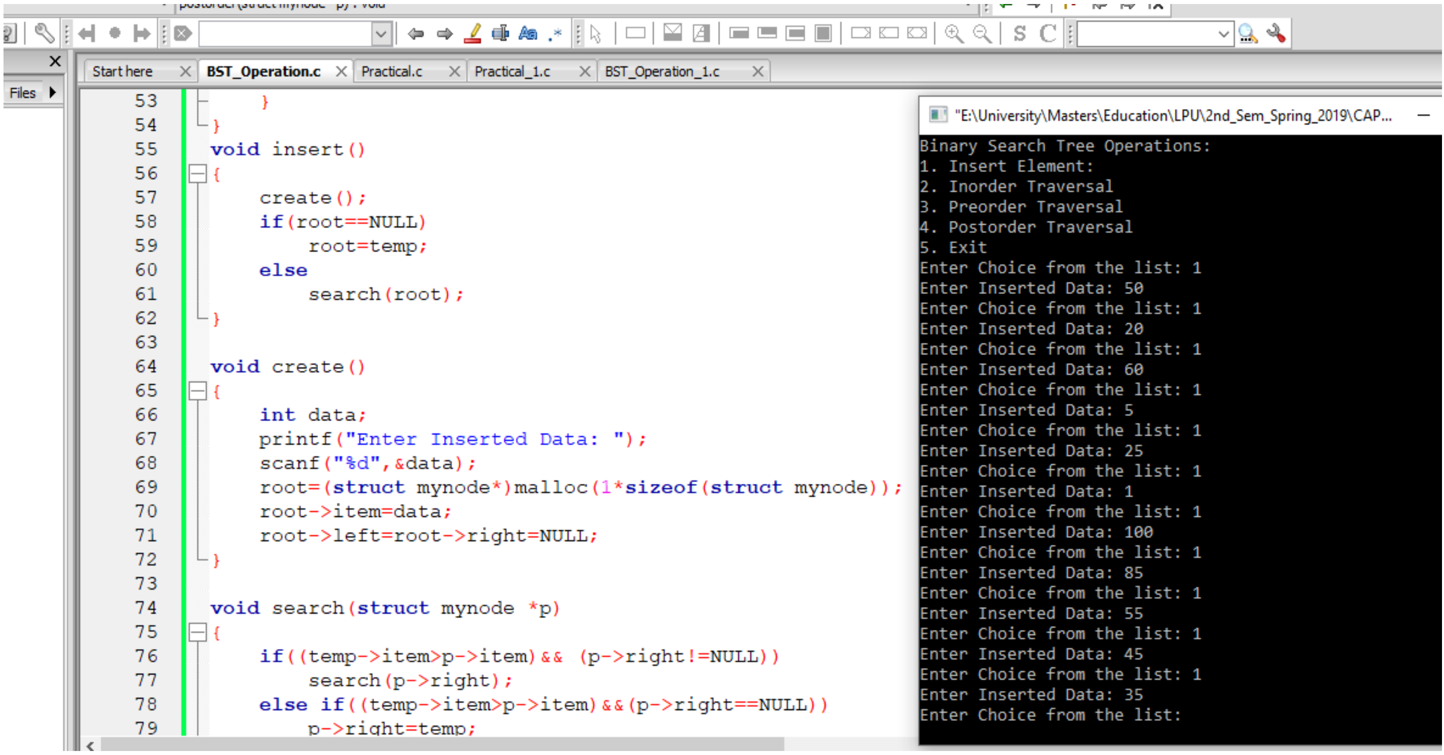
```
#include<stdio.h>
#include<stdlib.h>

struct mynode
{
   int item;
   struct mynode *left;
   struct mynode *right;
}*root = NULL,*temp=NULL;

void insert();
void create();
void search(struct mynode *p);
void inorder(struct mynode *p);
void preorder(struct mynode *p);
void postorder(struct mynode *p);
```

```
int flag=1;

void main()
{
    int choice;
    printf("\nBinary Search Tree Operations:");
    printf("\n1. Insert Element:\n");
    printf("2. Inorder Traversal\n");
    printf("3. Preorder Traversal\n");
    printf("4. Postorder Traversal\n");
    printf("5. Exit\n");
    while(1)
    {
        printf("\nEnter Choice from the list: ");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
            insert();
            break;
        case 2:
            inorder(root);
            break;
        case 3:
            preorder(root);
            break;
        case 4:
            postorder(root);
            break;
        case 5:
            exit(0);
        default :
            printf("Please Enter Correct Choice:  ");
            break;
        }
    }
}
void insert()
{
    create();
    if(root==NULL)
        root=temp;
    else
        search(root);
}

void create()
{
    int data;
    printf("Enter Inserted Data: ");
    scanf("%d",&data);
    root=(struct mynode*)malloc(1*sizeof(struct mynode));
    root->item=data;
    root->left=root->right=NULL;
}

void search(struct mynode *p)
{
    if((temp->item>p->item)&& (p->right!=NULL))
        search(p->right);
    else if((temp->item>p->item)&&(p->right==NULL))
        p->right=temp;
    else if((temp->item<p->item)&&(p->left!=NULL))
        search(p->left);
    else if((temp->item<p->item)&&(p->left==NULL))
        p->left=temp;
}

void inorder(struct mynode *p)
{
    if(root==NULL)
    {
        printf("Elements arn't available to display...");
        return;
    }
    if (p->left!=NULL)
        inorder(p->left);
    printf("%d->",p->item);
    if (p->right!=NULL)
        inorder(p->right);
```

```
}
void preorder(struct mynode *p)
{
    if (root==NULL)
    {
        printf("Elements arn't available to display...");
        return;
    }
    printf("%d->",p->item);
    if (p->left!=NULL)
        preorder(p->left);
    if (p->right!=NULL)
        preorder(p->right);
}
void postorder(struct mynode *p)
{
    if (root==NULL)
    {
        printf("Elements arn't available to display...");
        return;
    }
    if (p->left!=NULL)
        postorder(p->left);
    if (p->right!=NULL)
        postorder(p->right);
    printf("%d->",p->item);
}
```

Output:

Insert Operation:

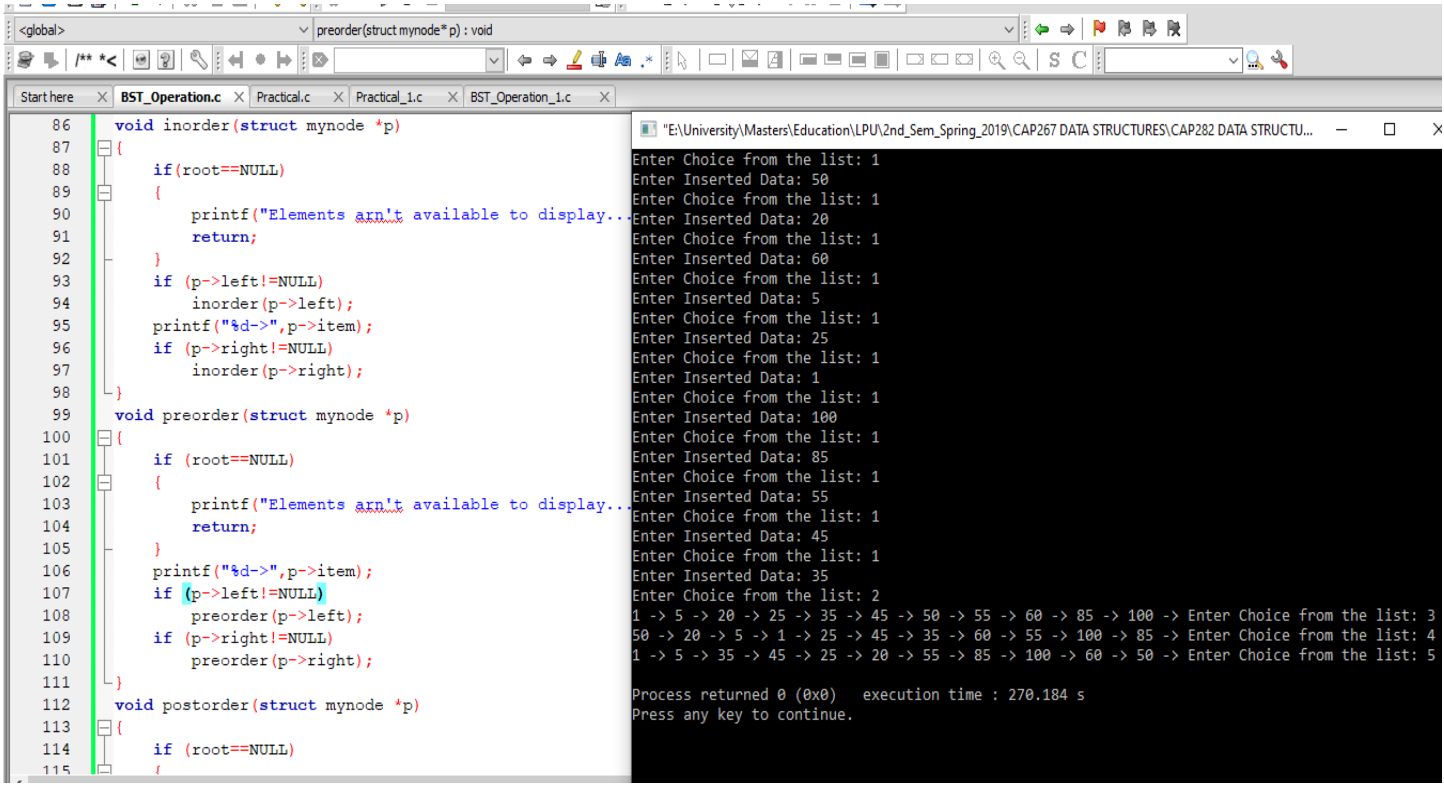


```
        }
}
void insert()
{
        create();
        if(root==NULL)
            root=temp;
        else
            search(root);
}

void create()
{
        int data;
        printf("Enter Inserted Data: ");
        scanf("%d",&data);
        root=(struct mynode*)malloc(1*sizeof(struct mynode));
        root->item=data;
        root->left=root->right=NULL;
}

void search(struct mynode *p)
{
        if((temp->item>p->item) && (p->right!=NULL))
            search(p->right);
        else if((temp->item>p->item)&&(p->right==NULL))
            p->right=temp;
```

"E:\University\Masters\Education\LPU\2nd_Sem_Spring_2019\CAP...

```
Binary Search Tree Operations:
1. Insert Element:
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Exit
Enter Choice from the list: 1
Enter Inserted Data: 50
Enter Choice from the list: 1
Enter Inserted Data: 20
Enter Choice from the list: 1
Enter Inserted Data: 60
Enter Choice from the list: 1
Enter Inserted Data: 5
Enter Choice from the list: 1
Enter Inserted Data: 25
Enter Choice from the list: 1
Enter Inserted Data: 1
Enter Choice from the list: 1
Enter Inserted Data: 100
Enter Choice from the list: 1
Enter Inserted Data: 85
Enter Choice from the list: 1
Enter Inserted Data: 55
Enter Choice from the list: 1
Enter Inserted Data: 45
Enter Choice from the list: 1
Enter Inserted Data: 35
Enter Choice from the list:
```

Traversing(Preorder,Inorder,Postorder) Operation:



```
void inorder(struct mynode *p)
{
    if(root==NULL)
    {
        printf("Elements arn't available to display...
        return;
    }
    if (p->left!=NULL)
        inorder(p->left);
    printf("%d->",p->item);
    if (p->right!=NULL)
        inorder(p->right);
}
void preorder(struct mynode *p)
{
    if (root==NULL)
    {
        printf("Elements arn't available to display...
        return;
    }
    printf("%d->",p->item);
    if (p->left!=NULL)
        preorder(p->left);
    if (p->right!=NULL)
        preorder(p->right);
}
void postorder(struct mynode *p)
{
    if (root==NULL)
    {
```

"E:\University\Masters\Education\LPU\2nd_Sem_Spring_2019\CAP267 DATA STRUCTURES\CAP282 DATA STRUCTU...

```
Enter Choice from the list: 1
Enter Inserted Data: 50
Enter Choice from the list: 1
Enter Inserted Data: 20
Enter Choice from the list: 1
Enter Inserted Data: 60
Enter Choice from the list: 1
Enter Inserted Data: 5
Enter Choice from the list: 1
Enter Inserted Data: 25
Enter Choice from the list: 1
Enter Inserted Data: 1
Enter Choice from the list: 1
Enter Inserted Data: 100
Enter Choice from the list: 1
Enter Inserted Data: 85
Enter Choice from the list: 1
Enter Inserted Data: 55
Enter Choice from the list: 1
Enter Inserted Data: 45
Enter Choice from the list: 1
Enter Inserted Data: 35
Enter Choice from the list: 2
1 -> 5 -> 20 -> 25 -> 35 -> 45 -> 50 -> 55 -> 60 -> 85 -> 100 -> Enter Choice from the list: 3
50 -> 20 -> 5 -> 1 -> 25 -> 45 -> 35 -> 60 -> 55 -> 100 -> 85 -> Enter Choice from the list: 4
1 -> 5 -> 35 -> 45 -> 25 -> 20 -> 55 -> 85 -> 100 -> 60 -> 50 -> Enter Choice from the list: 5

Process returned 0 (0x0)   execution time : 270.184 s
Press any key to continue.
```

Solution 2:

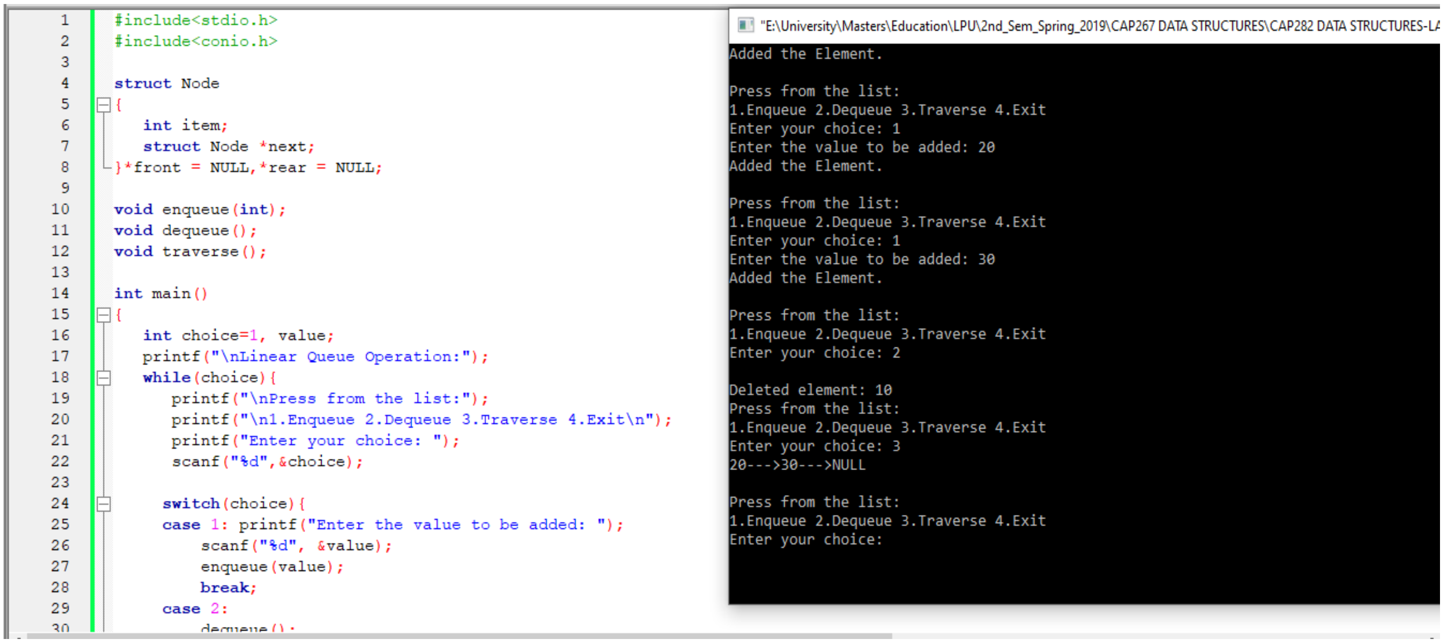
Source Code:

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

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

void enqueue(int);
void dequeue();
void traverse();

int main()
{
   int choice=1, value;
   printf("\nLinear Queue Operation:");
   while(choice){
      printf("\nPress from the list:");
      printf("\n1.Enqueue 2.Dequeue 3.Traverse 4.Exit\n");
      printf("Enter your choice: ");
      scanf("%d",&choice);

     switch(choice){
           case 1: printf("Enter the value to be added: ");
                     scanf("%d", &value);
                     enqueue(value);
                     break;
           case 2:
               dequeue();
               break;
           case 3:
               traverse();
               break;
           case 4:
               exit(0);
```

```
       default:
          printf("\nPlaese choose the correct input...\n");
    }
  }
  return 0;
}
void enqueue(int value)
{
  struct Node *newNode;
  newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->item = value;
  newNode -> next = NULL;
  if(front == NULL)
    front = rear = newNode;
  else{
    rear -> next = newNode;
    rear = newNode;
  }
  printf("Added the Element.\n");
}
void dequeue()
{
  if(front == NULL)
    printf("Queue is Empty.\n");
  else{
    struct Node *temp = front;
    front = front -> next;
    printf("\nDeleted element: %d", temp->item);
    free(temp);
  }
}
void traverse()
{
  if(front == NULL)
    printf("\nQueue is Empty!!!\n");
  else{
    struct Node *temp = front;
    while(temp->next != NULL){
       printf("%d--->",temp->item);
       temp = temp -> next;
```

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

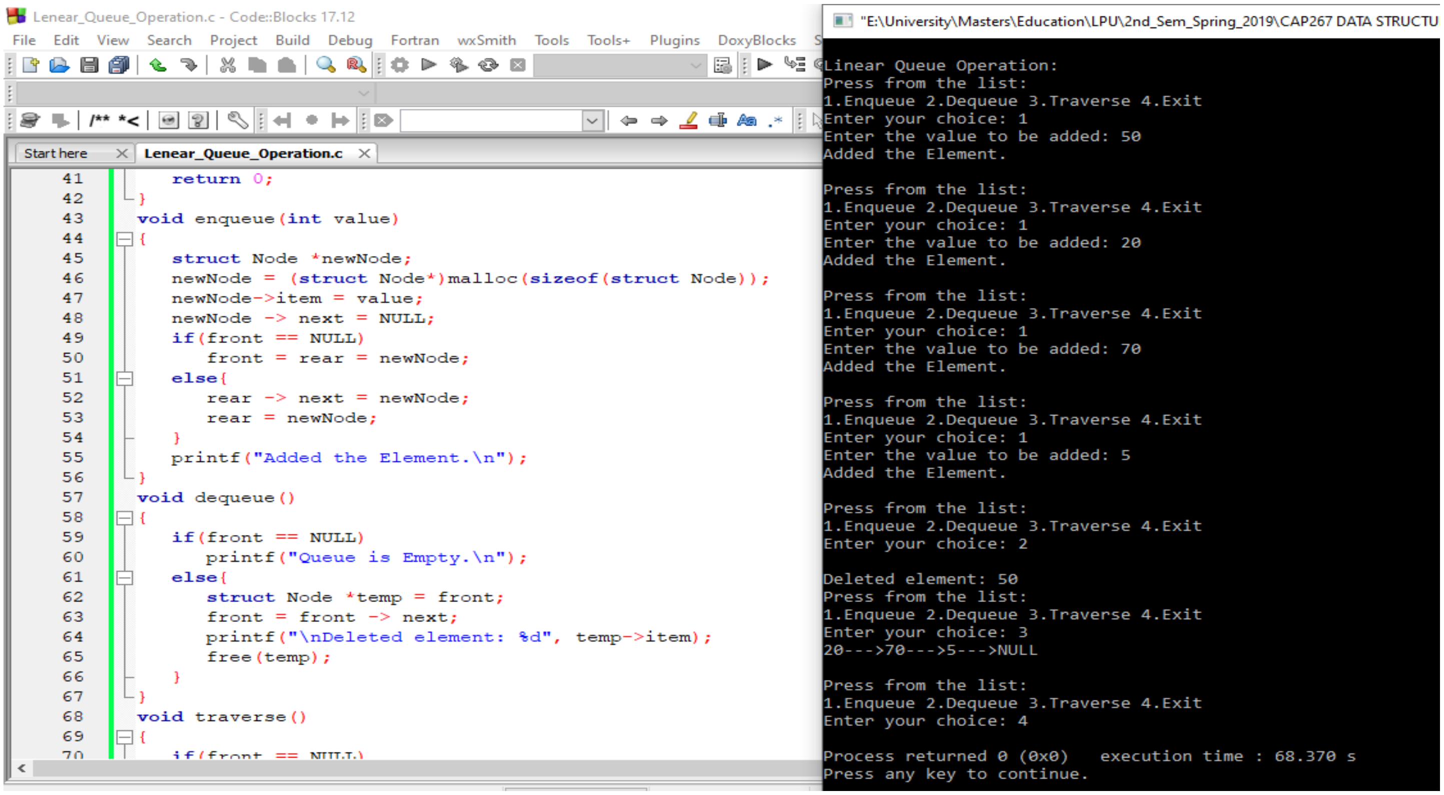
Output-1:


"E:\University\Masters\Education\LPU\2nd_Sem_Spring_2019\CAP267 DATA STRUCTURES\CAP282 DATA STRUCTURES-LA

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

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

void enqueue(int);
void dequeue();
void traverse();

int main()
{
    int choice=1, value;
    printf("\nLinear Queue Operation:");
    while(choice){
       printf("\nPress from the list:");
       printf("\n1.Enqueue 2.Dequeue 3.Traverse 4.Exit\n");
       printf("Enter your choice: ");
       scanf("%d",&choice);

      switch(choice){
      case 1: printf("Enter the value to be added: ");
          scanf("%d", &value);
          enqueue(value);
          break;
      case 2:
          dequeue();
```

```
Added the Element.

Press from the list:
1.Enqueue 2.Dequeue 3.Traverse 4.Exit
Enter your choice: 1
Enter the value to be added: 20
Added the Element.

Press from the list:
1.Enqueue 2.Dequeue 3.Traverse 4.Exit
Enter your choice: 1
Enter the value to be added: 30
Added the Element.

Press from the list:
1.Enqueue 2.Dequeue 3.Traverse 4.Exit
Enter your choice: 2

Deleted element: 10
Press from the list:
1.Enqueue 2.Dequeue 3.Traverse 4.Exit
Enter your choice: 3
20--->30--->NULL

Press from the list:
1.Enqueue 2.Dequeue 3.Traverse 4.Exit
Enter your choice:
```

Output-2:


Lenear_Queue_Operation.c - Code::Blocks 17.12
File
Edit
View
Search
Project
Build
Debug
Fortran
wxSmith
Tools
Tools+
Plugins
DoxyBlocks
Start here
Lenear_Queue_Operation.c
"E:\University\Masters\Education\LPU\2nd_Sem_Spring_2019\CAP267 DATA STRUCTU

```c
    return 0;
}
void enqueue(int value)
{
    struct Node *newNode;
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->item = value;
    newNode -> next = NULL;
    if(front == NULL)
       front = rear = newNode;
    else{
       rear -> next = newNode;
       rear = newNode;
    }
    printf("Added the Element.\n");
}
void dequeue()
{
    if(front == NULL)
       printf("Queue is Empty.\n");
    else{
       struct Node *temp = front;
       front = front -> next;
       printf("\nDeleted element: %d", temp->item);
       free(temp);
    }
}
void traverse()
{
    if(front == NULL)
```

```
Linear Queue Operation:
Press from the list:
1.Enqueue 2.Dequeue 3.Traverse 4.Exit
Enter your choice: 1
Enter the value to be added: 50
Added the Element.

Press from the list:
1.Enqueue 2.Dequeue 3.Traverse 4.Exit
Enter your choice: 1
Enter the value to be added: 20
Added the Element.

Press from the list:
1.Enqueue 2.Dequeue 3.Traverse 4.Exit
Enter your choice: 1
Enter the value to be added: 70
Added the Element.

Press from the list:
1.Enqueue 2.Dequeue 3.Traverse 4.Exit
Enter your choice: 1
Enter the value to be added: 5
Added the Element.

Press from the list:
1.Enqueue 2.Dequeue 3.Traverse 4.Exit
Enter your choice: 2

Deleted element: 50
Press from the list:
1.Enqueue 2.Dequeue 3.Traverse 4.Exit
Enter your choice: 3
20--->70--->5--->NULL

Press from the list:
1.Enqueue 2.Dequeue 3.Traverse 4.Exit
Enter your choice: 4

Process returned 0 (0x0)   execution time : 68.370 s
Press any key to continue.
```

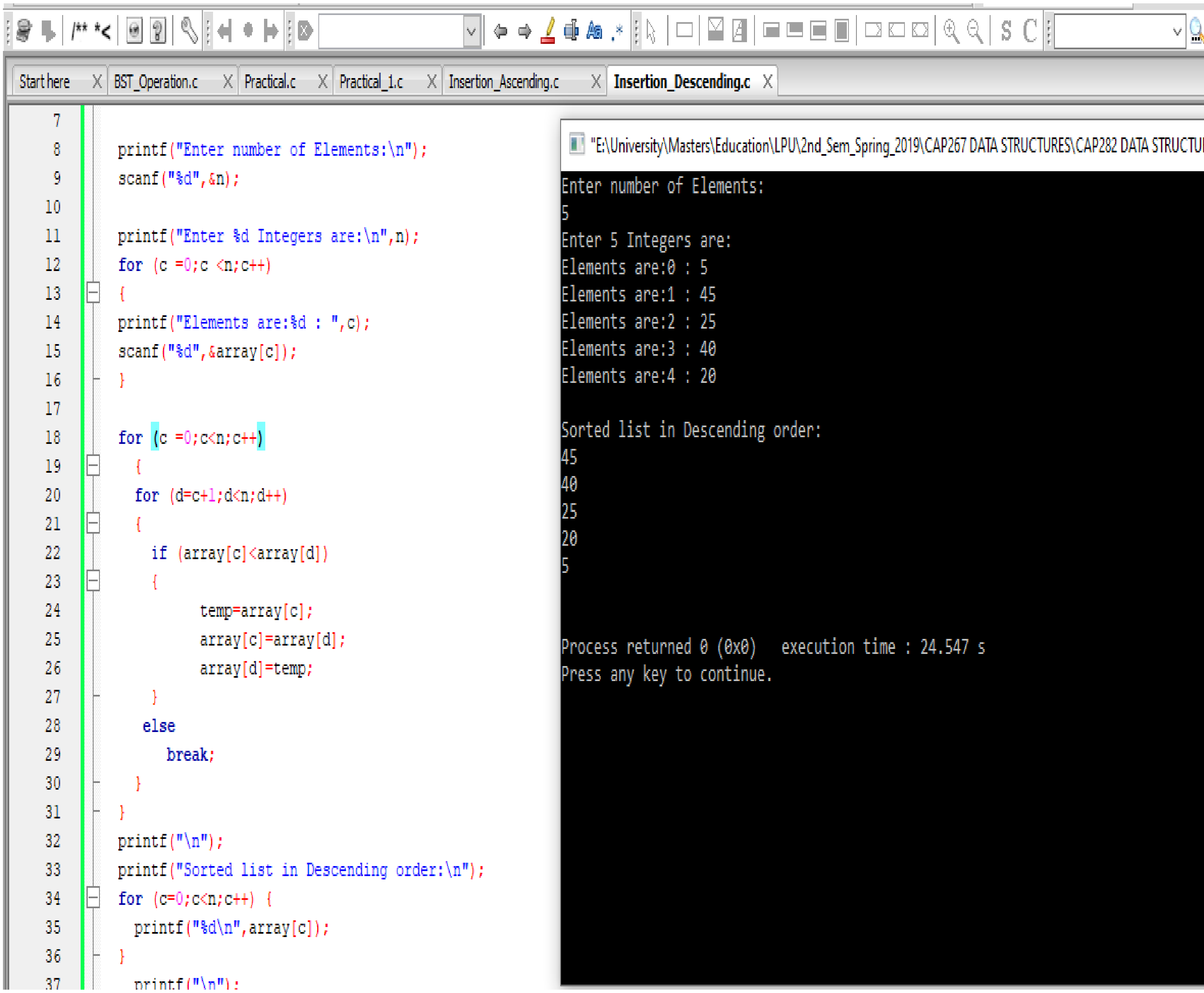
Solution 3:

Source Code:

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

int main()
{
  int n,array[1000],c,d,temp;

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

  printf("Enter %d Integers are:\n",n);
  for (c =0;c <n;c++)
  {
  printf("Elements are:%d : ",c);
  scanf("%d",&array[c]);
  }

  for (c =0;c<n;c++)
    {
    for (d=c+1;d<n;d++)
    {
      if (array[c]<array[d])
      {
            temp=array[c];
            array[c]=array[d];
            array[d]=temp;
      }
     else
        break;
    }
  }
  printf("\n");
  printf("Sorted list in Descending order:\n");
  for (c=0;c<n;c++) {
    printf("%d\n",array[c]);
  }
```

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

Output:



Start here
BST_Operation.c
Practical.c
Practical_1.c
Insertion_Ascending.c
Insertion_Descending.c
"E:\University\Masters\Education\LPU\2nd_Sem_Spring_2019\CAP267 DATA STRUCTURES\CAP282 DATA STRUCTU

```c

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

printf("Enter %d Integers are:\n",n);
for (c =0;c <n;c++)
{
printf("Elements are:%d : ",c);
scanf("%d",&array[c]);
}

for (c =0;c<n;c++)
  {
  for (d=c+1;d<n;d++)
  {
    if (array[c]<array[d])
    {
          temp=array[c];
          array[c]=array[d];
          array[d]=temp;
    }
   else
      break;
  }
}
printf("\n");
printf("Sorted list in Descending order:\n");
for (c=0;c<n;c++) {
  printf("%d\n",array[c]);
}
  printf("\n");
```

```
Enter number of Elements:
5
Enter 5 Integers are:
Elements are:0 : 5
Elements are:1 : 45
Elements are:2 : 25
Elements are:3 : 40
Elements are:4 : 20

Sorted list in Descending order:
45
40
25
20
5


Process returned 0 (0x0)   execution time : 24.547 s
Press any key to continue.
```