ONLINE SUBMISSION OF LAB RECORDS

Lab Subject	DATA STRUCTURES AND ALGORITHMS	**Submit Date**	30-06-2020
SIC No	190610193	**Regd. No**	1901209198
Name	ANIRUDH PANDA	**Branch**	EEE
Semester	2ND	**Section/Group**	D/D2

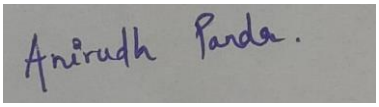
INDEX

UNDERTAKING

I hereby declare that, I had submitted the laboratory records for the experiments which were physically completed in the Institute and the lab record folder is properly preserved by me. In this online submission, I am submitting the lab records for the experiments (as mentioned in the index above) which were explained and demonstrated in online mode. I have written the lab record myself, scanned into PDF format for soft copy submission. I undertake that I will preserve these hard copies with me. After the institute reopens, I will add these pages to the existing lab record folder and submit the complete folder within 7 days of reopening. I understand that, unless I submit the complete lab record folder in hard copy form, the marks awarded in the lab subject may be revoked by the institute.

Date: 30-06-2020

Full Signature of Student



ASSIGNMENT-5

1 –

*/*Write a program to create a linked list and traverse all elements*/*

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

struct node
{
    int data;
    struct node *next;
} * start, *current, *newnode;

void create();
void display();
void main()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d", &x);
    newnode->data = x;
    newnode->next = start;
    start = newnode;
}

void display()
{
    struct node *p;
    p = start;
    while (p->data != NULL)
    {
        printf("%d", p->data);
        p = p->next;
    }
}
```

2 –

*/*Write a menu driven program to implement single linked list to perform the following operations: 1- Insertion at Beginning 2- Delete from Beginning 3- Display 4- Exit */*

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

```
#include <malloc.h>

struct node
{
    int data;
    struct node *next;
} * start, *current, *newnode;

void main()
{
    int x,ch;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d", &x);
    newnode->data = x;
    newnode->next = start;
    start = newnode;
}
while(1)
{
    printf("1- Insert at beggining \n 2- Delete from beggining \n 3- Display \n 4- Exit \n");
    printf("Enter your choice");
    scanf("%d",&ch);

    switch (ch)
    {
    case 1:
        insertbeg();
        break;
    case 2:
        deletebeg();
        break;

    case 3:
       display();
        break;

    case 4:
        exit();

    default:
        printf("Invalid input");
        break;
    }
}

void insertbeg()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d",&x);
    newnode->data=x;
```

```
    newnode->next=start;
    start=newnode;
}

void deletebeg()
{
    struct node *p;
    p=start;
    start=start->next;
    free(p);
}

void display()
{
    struct node *p;
    p = start;
    while (p->data != NULL)
    {
        printf("%d", p->data);
        p = p->next;
    }
}
```

3-

*/*Write separate user defined functions to perform the following operations on a single linked list: (i) Insertion at End (ii) Insertion after a specific value (iii) Insert before a specific value */*

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

struct node
{
    int data;
    struct node *next;
} * start, *current, *newnode;
void insertend();
void insertafter();
void insertbefore();

void main()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d", &x);
    newnode->data = x;
    newnode->next = start;
    start = newnode;
}
```

```
void insertend()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d",&x);
    newnode->data=x;
    newnode->next=NULL;
    current->next=newnode;
    current=newnode;
}

void insertafter()
{
    int x,ele;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d",&x);
    printf("Enter the value after which you want to insert");
    scanf("%d", &ele);
    struct node *p,*q;
    p=start;
    while(p->data!=ele)
    {
        p=p->next;
    }
    q=p->next;
    newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data=x;
    p->next=newnode;
    newnode->next=q;
}

void insertbefore()
{
    int x,ele;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d",&x);
    printf("Enter the value after which you want to insert");
    scanf("%d", &ele);
    struct node *p,*q;
    while(p->next->data!=ele)
    {
        p=p->next;
    }
    q=p->next;
    newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data=x;
    p->next=newnode;
    newnode->next=q;
}
```

4-

*/*Write separate user defined functions to perform the following operations on a single linked list: (i) Delete from End (ii) Delete after a specific value (iii) Delete a specific data value (iv) Delete a node before a value */*

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

struct node
{
    int data;
    struct node *next;
} * start, *current, *newnode;
void deleteend();
void deleteafter();
void deletespecific();
void deletenodebefore();

void main()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d", &x);
    newnode->data = x;
    newnode->next = start;
    start = newnode;
}

void deleteend()
{
    struct node *p,*q;
    p=start;
    while(p->next!=current)
    {
        p=p->next;
    }
    q=p->next;
    p->next=NULL;
    current=p;
    free(q);
}

void deleteafter()
{
    struct node *p,*q;
    int ele;
    printf("Enter the specific element");
    scanf("%d", &ele);
    while(p->data!=ele)
```

```
    {
        p=p->next;
    }
    q=p->next;
    p->next=q->next;
    free(q);
}

void deletespecific()
{
    struct node *p,*q;
    int ele;
    printf("Enter the specific element");
    scanf("%d", &ele);
    while(p->next->data!=ele)
    {
        p=p->next;
    }
    q=p->next;
    p->next=q->next;
    free(q);
}

void deletenodebefore()
{
    struct node *p,*q;
    int ele;
    printf("Enter the specific element");
    scanf("%d", &ele);
    while(p->next->next->data!=ele)
    {
        p=p->next;
    }
    q=p->next;
    p->next=q->next;
    free(q);
}
```

5-

*/*Write a user defined function to count the number of nodes present in a single linked list. */*

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

struct node
{
    int data;
    struct node *next;
} * start, *current, *newnode;
```

```
void main()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d", &x);
    newnode->data = x;
    newnode->next = start;
    start = newnode;
}

void count()
{
    struct node *p;
    p=start;
    int c=1;
    while(p->next!=NULL)
    {
        p=p->next;
        c++;
    }
    printf("The number of nodes are %c",c);
}
```

6-

*/*Write separate user defined functions to perform the following operations on a single linked list: (i) Insert after ith node*/*

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

struct node
{
    int data;
    struct node *next;
} * start, *current, *newnode;
void insertafteri();

void main()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d", &x);
    newnode->data = x;
    newnode->next = start;
    start = newnode;
}

void insertafteri()
```

```
{
    int x,i,pos;
    struct node *p;
    p=start;
    i=0;
    printf("Enter the node position");
    scanf("%d",&pos);
    while(p->next!=NULL && i<pos)
    {
        p=p->next;
        i++;
    }
     newnode = (struct node *)malloc(sizeof(struct node));
     printf("Enter data for new node");
     scanf("%d",&x);
     newnode->data=x;
     newnode->next=p->next;
     p->next=newnode;
}
```

ASSIGNMENT-6

1-

*/*Write a user defined function to Search an element in a single linked list*/*

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

struct node
{
    int data;
    struct node *next;
} * start, *current, *newnode;

void search();
void main()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d", &x);
    newnode->data = x;
    newnode->next = start;
    start = newnode;
}
```

```
void search()
{
    int ele;
    struct node *p;
    p=start;
    while(p->data!=ele)
    {
        p=p->next;
    }
    printf("%d", p->data);
}
```

2-

*/*Write a user defined function to Sort the list in ascending order. */*

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

struct node
{
    int data;
    struct node *next;
} * start, *current, *newnode;

void sort();
void main()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d", &x);
    newnode->data = x;
    newnode->next = start;
    start = newnode;
}

void sort()
{
    struct node *p,*q;
    int temp;
    for(p=start;p->next!=NULL;p=p->next)
    for(q=p->next;q!=NULL;q=q->next)
    {
        if (p->data>q->data)
        {
            temp=p->data;
            p->data=q->data;
            q->data=temp;
        }
```

```
    }
}
```

3-

*/*Write user defined functions to find the smallest and largest element in a single linked list*/*

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

struct node
{
    int data;
    struct node *next;
} * start, *current, *newnode;

void largest();
void smallest();
void main()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d", &x);
    newnode->data = x;
    newnode->next = start;
    start = newnode;
}

void largest()
{
    struct node *p;
    p=start;
    int max;
    max=p->data;
    while(p!=NULL)
    {
        if(p->data>max)
        max=p->data;
        p=p->next;

    }
}

void smallest()
{
    struct node *p;
    p=start;
    int min;
    min=p->data;
    while(p!=NULL)
    {
```

```
        if(p->data<min)
        min=p->data;
        p=p->next;
    }
}
```

4-

*/*Write a user defined function to reverse the list. */*

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

struct node
{
    int data;
    struct node *next;
} * start, *current, *newnode;

void reverse();
void main()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d", &x);
    newnode->data = x;
    newnode->next = start;
    start = newnode;
}

void reverse()
{
    struct node *p,*c,*n;
    p=NULL;
    c=start;
    while(c!=NULL)
    {
        n=c->next;
        c->next=p;
        p=c;
        c=n;
    }
}
```

5-

/*Write a program to create two separate linked lists and merge the elements*/

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

struct node
{
    int data;
    struct node *next;
} * start, *current, *newnode;

void merge();
void main()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d", &x);
    newnode->data = x;
    newnode->next = start;
    start = newnode;
}

void merge()
{
    struct node *p, *q, *s;

    if (p->data <= q->data)
    {
        s = p;
        s->next = (p->next, q);
    }
    else
    {
        s = q;
        s->next = (p, q->next);
    }
    return (s);
}
```

6-

/*Implement stack using linked list representation to perform the following operations: 1-Push, 2-Pop, 3-Display, 4-Exit*/

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

struct node
{
    int data;
    struct node *next;
```

```
} *top=NULL,*newnode;

void push()
{
    int x;
    newnode=(struct node *) malloc (sizeof (struct node));
    printf("Enter the value of x");
    scanf("%d",&x);
    newnode->data=x;
    newnode->next=top;
    top=newnode;
}

void pop()
{
    struct node *p;
    p=top;
    printf("%d is deleted from stack", top->data);
    top=top->next;
    free(p);
}

void display()
{
    struct node *p;
    p=top;
    while(p!=NULL)
    {
        printf("%d",p->data);
        p=p->next;
    }
}
```

7-

/*Implement Linear Queue using linked list representation to perform the following operations: 1-Insert, 2-Delete, 3-Display, 4-Exit*/

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

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

void insert()
{
    int x;
    newnode=(struct node *) malloc (sizeof (struct node));
```

```
    printf("Enter the value of x");
    scanf("%d",&x);
    newnode->data=x;
    newnode->next=NULL;
    if(rear==NULL)
    {
       front=newnode;
       rear=newnode;
    }
    else
    {
       front=newnode;
       front->next=rear;
    }

}

void delete()
{
    struct node *p;
    p=front;
    front=front->next;
    printf("%d is deleted",p->data);
}

void display()
{
    struct node *p;
    p=front;
    while(p!=NULL)
    {
       printf("%d",p->data);
       p=p->next;
    }
}
```

ASSIGNMENT-7

1-

*/*create and traverse the element in double linked list*/*

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

```
 #include <malloc.h>
 struct node  {      int data;      struct node *next, *prev;  } * start, *current, *newnode;
 void create();
 void traverseforward();
 void traversebackward();
 void main()
{      int x;
newnode= (struct node*)malloc(sizeof(struct node));
printf("Enter the value of x");
scanf("%d", &x);
newnode->data = x;
newnode->prev= NULL;
newnode->next=start;
start->prev=newnode;
start=newnode;  }
void traverseforward()
{      struct node *p;
p=start;
while(p!= NULL)
{
printf("%d",p->data);
  p=p->next;
 }
}    void traversebackward()
{      struct node *p;
p=current;
while(p!= NULL)
{
printf("%d",p->data);
 p=p->prev;
}
}
```

2-

*/*Write a menu driven program to implement a double linked list with the following options: 1-Insert at Beginning 2-Display 3-Delete from Beginning 4-Exit*/*

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

struct node
{
    int data;
    struct node *next, *prev;
} * start, *newnode, *current;

main()
{
    int x, ch;
```

```
    printf("Enter the data of the 1st node");
    scanf("%d", &x);
    newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data = x;
    newnode->prev = NULL;
    newnode->next = NULL;
    start = newnode;
    current = newnode;
}
while (1)
{
    printf("1- Insert at beggining \n 2- Display \n 3- Delete at beggining \n 4- Exit \n");
    printf("Enter your choice");
    scanf("%d", &ch);

    switch (ch)
    {
    case 1:
        insertbeg();
        break;
    case 2:
        display();
        break;

    case 3:
       insertend(();
        break;

    case 2:
        exit();

    default:
        printf("Invalid input");
        break;
    }
}

void insertbeg()
{
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d", &x);
    newnode->data = x;
    newnode->prev = NULL;
    newnode->next = start;
    start->prev = newnode;
    start = newnode;
}

void display()
{
    struct node *p;
```

```
    p = start;
    while (p != NULL)
    {
        printf("%d \t", p->data);
        p = p->next;
    }
}

void deletebeg()
{
    struct node *p;
    p = start;
    start = start->next;
    free(p);
    start->prev = NULL;
}
```

3-

*/*Write separate user defined functions for the following insert operations in a double linked list (i) Insert at End (ii) Insert after a specific value (iii) Insert before a specific value */*

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

struct node
{
    int data;
    struct node *next, *prev;
} * start, *newnode, *current;

void insertend();
void insertafterspecific();
void insertbeforespecific();

main()
{
    int x, ch;
    printf("Enter the data of the 1st node");
    scanf("%d", &x);
    newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data = x;
    newnode->prev = NULL;
    newnode->next = NULL;
    start = newnode;
    current = newnode;
}

void insertend()
{
```

```
    int x;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d",&x);
    newnode->data=x;
    newnode->next=NULL;
    newnode->prev=current;
    current->next=newnode;
    current=newnode;
}

void insertafterspecific()
{
    int x,ele;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d",&x);
    printf("Enter the value after which you want to insert");
    scanf("%d", &ele);
    struct node *p,*q;
    p=start;
    while(p->data!=ele)
    {
        p=p->next;
    }
    q=p->next;
    newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data=x;
    newnode->prev=p;
    p->next=newnode;
    newnode->next=q;
    q->prev=newnode;
}

void insertbeforespecific()
{
    int x,ele;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the value of x");
    scanf("%d",&x);
    printf("Enter the value after which you want to insert");
    scanf("%d", &ele);
    struct node *p,*q;
    while(p->next->data!=ele)
    {
        p=p->next;
    }
    q=p->next;
    newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data=x;
    newnode->prev=p->prev;
    p->prev=newnode;
    newnode->next=q;
```

```
    q->prev->next=newnode;
}
```

4-

*/*Write separate user defined functions for the following Delete operations in a double linked list (i) Delete from End (ii) Delete after a specific value (iii) Delete before a specific value (iv) Delete a specific value */*

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

struct node
{
    int data;
    struct node *next, *prev;
} * start, *newnode, *current;

void deleteend();
void deleteafterspecific();
void deletebeforespecific();
void deletespecific();
main()
{
    int x, ch;
    printf("Enter the data of the 1st node");
    scanf("%d", &x);
    newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data = x;
    newnode->prev = NULL;
    newnode->next = NULL;
    start = newnode;
    current = newnode;
}

void deleteend()
{
    struct node *p;
    p = current;
    curent = current->prev;
    free(p);
    current->next = NULL;
}

void deleteafterspecific()
{
    struct node *p, *q, *r;
    p = start;
    int ele;
    printf("Enter the specific element");
    scanf("%d", &ele);
    while(p->data!=ele)
    {
```

```
        p=p->next;
    }
    q=p->next;
    r=q->next;
    p->next=r;
    r->prev=p;
    free(q);
}

void deletebeforespecific()
{
    struct node *p,*q,*r;
    p=start;
    int ele;
    printf("Enter the specific element");
    scanf("%d", &ele);
    while(p->data!=ele)
    {
        p=p->next;
    }
    q=p->prev;
    r=q->prev;
    r->next=p;
    p->prev=r;
    free(q);
}

void deletespecific()
{
    int ele;
    printf("Enter the element to be deleted");
    scanf("%d",&ele);
    struct node *p,*q,*r;
    p=start;
    while(p->data!=ele)
    {
        p=p->next;
    }
    q=p->next;
    r=p->prev;
    r->next=q;
    q->prev=r;
    free(p);
}
```

ASSIGNMENT-8

1-

*/*Write a menu driven program with the following options to construct a binary search tree (BST) recursively and traverse the elements:*/*

1-Insert
2-Pre-orded traversal
3-In order traversal
4-Post order traversal
5-Exit

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

struct node
{
   int data;
   struct node *left;
   struct node *right;
} *root = NULL;

struct node *insert(struct node *, int);
void preorder(struct node *);
void inorder(struct node *);
void postorder(struct node *);

main()
{
   int ch, x;
   while (1)
   {
      printf("\nMenu: \n1: insert\n2: pre-order traversal\n 3: in-order traversal\n 4: post-order traversal\n \n 5: exit\n");
      printf("\n Enter your choice");
      scanf("%d", &ch);
      switch (ch)
      {
      case (1):
         printf("enter the data to insert:");
         scanf("%d", &x);
         root = insert(root, x);
         break;
      case (2):
         preorder(root);
         break;
      case (3):
         inorder(root);
```

```
        break;
      case (4):
        postorder(root);
        break;
      case (5):
        exit(0);
      default:
        printf("Invalid option");
      }
   }
}

struct node *insert(struct node *temp, int ele)
{
   if (temp == NULL)
   {
      temp = (struct node *)malloc(sizeof(struct node));
      temp->data = ele;
      temp->left = NULL;
      temp->right = NULL;
   }
   else
   {
      if (ele < temp->data)
         temp->left = insert(temp->left, ele);
      else
      {
         if (ele > temp->data)
            temp->right = insert(temp->right, ele);
      }
   }
   return temp;
}

void preorder(struct node *ptr)
{
   if (ptr != NULL)
   {
      printf("%d\t", ptr->data);
      preorder(ptr->left);
      preorder(ptr->right);
   }
}
void inorder(struct node *ptr)
{
   if (ptr != NULL)
   {
      inorder(ptr->left);
      printf("%d\t", ptr->data);
      inorder(ptr->right);
   }
}
void postorder(struct node *ptr)
```

```
{
   if (ptr != NULL)
   {
      postorder(ptr->left);
      postorder(ptr->right);
      printf("%d\t", ptr->data);
   }
}
```

2-

Write a menu driven program to perform the following operations on a BST:

1-Insert
2- In-order traversal
3- Search an element
4- Find minimum
5- Find maximum
6- Exit

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

struct node
{
    int data;
    struct node *left;
    struct node *right;
} *root = NULL;

struct node *insert(struct node *, int);
void inorder(struct node *);
struct node *search(struct node *, int);
struct node *min(struct node *temp);
struct node *max(struct node *temp);

main()
{
    int ch, x, val;
    while (1)
    {
        printf("\nMenu: \n1: insert\n2: in-order traversal\n 3: Search\n 4:  Minimum\n \n 5: Maximum\n 6:
exit\n");
        printf("\n Enter your choice");
        scanf("%d", &ch);
        switch (ch)
        {
        case (1):
```

```
            printf("enter the data to insert:");
            scanf("%d", &x);
            root = insert(root, x);
            break;
        case (2):
            inorder(root);
            break;
        case (3):
            printf("Enter element to be searched");
            scanf("%d", &val);
            root = search(root, val);

        case (4):
            min(root);
            break;
        case (5):
            max(root);
            break;
        case (6):
            exit(0);
            break;
        default:
            printf("Invalid option");
        }
    }
}

struct node *insert(struct node *temp, int ele)
{
    if (temp == NULL)
    {
        temp = (struct node *)malloc(sizeof(struct node));
        temp->data = ele;
        temp->left = NULL;
        temp->right = NULL;
    }
    else
    {
        if (ele < temp->data)
            temp->left = insert(temp->left, ele);
        else
        {
            if (ele > temp->data)
                temp->right = insert(temp->right, ele);
        }
    }
    return temp;
}

void inorder(struct node *p)
{
    if (p != NULL)
    {
```

```
        inorder(p->left);
        printf("%d \t", p->data);
        inorder(p->right);
    }
}

struct node *search(struct node *temp, int val)
{
    struct node *p;
    p = temp;
    if (p != NULL && p->data != val)
    {
        if (val < p->data)
        {
            p = p->left;
        }

        else
        {
            if (val > p->data)
                p = p->right;
        }
    }
    if (p == NULL)
    {
        printf("Element not found");
    }
    else
    {
        return p;
    }
}

struct node *min(struct node *temp)
{
    if (temp == NULL)
        return NULL;
    if (temp->left == NULL)
    {
        return temp;
    }
    else
    {
        return (min(temp->left));
    }
}

struct node *max(struct node *temp)
{
    if (temp == NULL)
        return NULL;
    if (temp->right == NULL)
    {
```

```
        return temp;
    }
    else
    {
        return (max(temp->right));
    }
}
```

3-

Write a menu driven program to perform the following operations on a BST:
1-Insert
2- In-order traversal
3-Delete
4-Exit

-

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

struct node
{
    int data;
    struct node *left;
    struct node *right;
} *root = NULL;

struct node *insert(struct node *, int);
void inorder(struct node *);
struct node *delete (struct node *temp, int val);

main()
{
    int ch, x, val;
    while (1)
    {
        printf("\nMenu: \n1: insert\n2: in-order traversal\n 3: Delete\n  4: exit\n");
        printf("\n Enter your choice");
        scanf("%d", &ch);
        switch (ch)
        {
        case (1):
            printf("enter the data to insert:");
            scanf("%d", &x);
            root = insert(root, x);
            break;
        case (2):
            inorder(root);
            break;
        case (3):
```

```
            printf("Enter element to be searched");
            scanf("%d", &val);
            root = delete (root, val);

        case (4):
            exit(0);
            break;
        default:
            printf("Invalid option");
        }
    }
}

struct node *insert(struct node *temp, int ele)
{
    if (temp == NULL)
    {
        temp = (struct node *)malloc(sizeof(struct node));
        temp->data = ele;
        temp->left = NULL;
        temp->right = NULL;
    }
    else
    {
        if (ele < temp->data)
            temp->left = insert(temp->left, ele);
        else
        {
            if (ele > temp->data)
                temp->right = insert(temp->right, ele);
        }
    }
    return temp;
}

void inorder(struct node *p)
{
    if (p != NULL)
    {
        inorder(p->left);
        printf("%d \t", p->data);
        inorder(p->right);
    }
}

struct node *delete (struct node *temp, int val)
{
    if (temp == NULL)
        return temp;

    if (val < temp->data)
    {
        temp->left = delete (temp->left, val);
```

```
    }
    else
    {
        if (val >> temp->data)
        {
            temp->right = delete (temp->right, val);
        }
        else
        {
            if (temp->left == NULL)
            {
                struct node *p = temp->right;
                free(temp);
                return p;
            }
            else if (temp->right == NULL)
            {
                struct node *p = temp->left;
                free(temp);
                return p;
            }
        }
    }
}
```

ASSIGNMENT-9

1-

*/*Implement bubble sort to sort the elements of any user entered array in ascending order. */*

```
#include <stdio.h>
void bubblesort(int[], int);
main()
{
    int a[20], n, i;
    printf("Enter the number of elements in the array");
    scanf("%d", &n);
    printf("Enter the array elements");
    for (i = 0; i < n; i++)
    {
        scanf("%d\t", &a[i]);
    }
    printf("The unsorted array is ", a[i]);
```

```
}

void bubblesort(int a[], int n)
{
    int i, j, temp;
    for (i = 0; i < n - 1; i++)
    {
        for (j = 0; j < n - 1 - i; i++)
        {
            if (a[j] > a[j + 1])
            {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }
    printf("Sorted list in ascending order:\n");

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

2-

*/*Implement selection sort to sort the elements of any user entered array in ascending order. */*

```
#include <stdio.h>
void selectionsort(int[], int);

main()
{
    int a[20], i, n;
    printf("Enter the number of elements in the array");
    scanf("%d", &n);
    printf("Enter the array elements");
    for (i = 0; i < n; i++)
    {
        scanf("%d \t", &a[i]);
    }
    printf("The unsorted array is %d", a[i]);
}

void selectionsort(int a[], int n)
{
    int i, j, min, temp;
    for (i = 0; i = n - 1; i++)
    {
        min = i;
```

```
        for (j = i + 1; j < n; j++)
        {
            if (a[min] > a[j])
                ;
            min = j;
        }
        if (min != i)
        {
            temp = a[i];
            a[i] = a[min];
            a[min] = temp;
        }
    }
    printf("Sorted list in ascending order:\n");

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

    return 0;
}
```

3-

*/*Implement insertion sort to sort the elements of any user entered array in ascending order. */*

```
#include <stdio.h>

void insertionsort(int[], int);

main()
{
    int a[20], i, n;
    printf("Enter the number of elements in the array");
    scanf("%d", &n);
    printf("Enter the array elements");
    for (i = 0; i < n; i++)
    {
        scanf("%d \t", &a[i]);
    }
    printf("The unsorted array is %d", a[i]);
}

void insertionsort(int a[], int n)
{
    int key, i, j, ;
    for (i = 1; i < n; i++)
    {
        key = a[i];
        j = i - 1;
        while (j >= 0 && a[j] > key)
```

```
        {
            a[j + 1] = a[j];
            j = j - 1;
        }
        a[j + 1] = key;
    }
}
```

4-

*/*Implement merge sort to sort the elements of any user entered array in ascending order*/*

```
#include<stdio.h>

void mergesort(int a[],int i,int j);
void merge(int a[],int i1,int j1,int i2,int j2);

int main()
{
        int a[30],n,i;
        printf("Enter no of elements:");
        scanf("%d",&n);
        printf("Enter array elements:");

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

        mergesort(a,0,n-1);

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

        return 0;
}

void mergesort(int a[],int i,int j)
{
        int mid;

        if(i<j)
        {
                mid=(i+j)/2;
                mergesort(a,i,mid);
                mergesort(a,mid+1,j);
                merge(a,i,mid,mid+1,j);
        }
}

void merge(int a[],int i1,int j1,int i2,int j2)
{
        int temp[50];
```

```
        int i,j,k;
        i=i1;
        j=i2;
        k=0;

        while(i<=j1 && j<=j2)
        {
                if(a[i]<a[j])
                        temp[k++]=a[i++];
                else
                        temp[k++]=a[j++];
        }

        while(i<=j1)
                temp[k++]=a[i++];

        while(j<=j2)
                temp[k++]=a[j++];


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

5-

*/*Implement quick sort to sort the elements of any user entered array in ascending order*/*

```
#include <stdio.h>
void quicksort(int a[], int p, int r);
int partition(int a[], int p, int r);

int main()
{
   int a[30], n, i;
   printf("Enter no of elements:");
   scanf("%d", &n);
   printf("Enter array elements:");

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

   quicksort(a, 0, n - 1);

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

   return 0;
}
```

```
void quicksort(int a[], int p, int r)
{
    int q;
    if (p < r)
    {
        q = partition(a, p, r);
        quicksort(a, p, q);
        quicksort(a, q + 1, r);
    }
}

int partition(int a[], int p, int r)
{
    int pivot, i, j, temp;
    pivot = a[p];
    i = p - 1;
    j = r + 1;
    while (1)
    {
        do
        {
            j = j - 1;
        } while (a[j] > pivot);
        do
        {
            i = i + 1;
        } while (a[i] < pivot);
        if (i < j)
        {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
        else
            return (j);
    }
}
```