EXPERIMENT 1.

/* Design, develop and implement a menu driven program in c for the following array operations.

a) inserting an element (ELEM) at a given valid position (POS)

b) deleting an element at a given valid position.

c) display of array elements

d) exit.

Support the program with functions for each of the above operations. */

```
#include<stdio.h>
#include<stdlib.h>
int a[10];
int n, ELEM, POS, i;
void display();
void insert();
void del();

int main()
{

    int choice=1;
    printf("Enter the size of array\n");
    scanf("%d",&n);
    printf("Enter the elements of array\n");
    for(i=0;i<n;i++)
     scanf("%d",&a[i]);
    while(choice)
    {
        printf("\n\n........MENU.........\n\n");
        printf(" 1. Display\n 2. Insert\n 3. delete\n 4.Exit\n");
        printf("Enter your choice:  ");
        scanf("%d",&choice);
        switch(choice)
        {
           case 1: display();
                  break;
            case 2: insert();
                  break;
            case 3: del();
                  break;
            case 4: exit(0);
            default: printf("Invalid Choice\n");
        }

    }
    return 0;
}

void display()
{
    printf("The array elements are \n");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);

    }

}
```

```
   void insert()
   {
       printf("Enter the position for new element : \n");
       scanf("%d",&POS);
       printf("Enter the element to be inserted: \n");
       scanf("%d",&ELEM);
       for(i=n-1;i>=POS;i--)
        a[i+1]=a[i];
       a[POS]=ELEM;
       n=n+1;
   }

   void del()
   {
       printf("Enter position of element to be deleted:\n");
       scanf("%d",&POS);
       ELEM=a[POS];
       for(i=POS;i<n-1;i++)
        a[i]=a[i+1];
       n=n-1;
       printf("Deleted element is %d\n",ELEM);
   }


   /* OUPUT:
   Enter the size of array
3
Enter the elements of array
10
20
```

```
30



........MENU.........

 1. Display
 2. Insert
 3. delete
 4.Exit
Enter your choice:  1
The array elements are
10      20      30

........MENU.........

 1. Display
 2. Insert
 3. delete
 4.Exit
Enter your choice:  2
Enter the position for new element :
1
Enter the element to be inserted:
2222



........MENU.........

 1. Display
 2. Insert
 3. delete
 4.Exit
Enter your choice:  1
The array elements are
10      2222    20      30

........MENU.........

 1. Display
 2. Insert
 3. delete
 4.Exit
Enter your choice:  3
Enter position of element to be deleted:
1
Deleted element is 2222


........MENU.........

 1. Display
 2. Insert
 3. delete
 4.Exit
Enter your choice:  1
The array elements are
10      20      30

........MENU.........
```

```
 1. Display
 2. Insert
 3. delete
 4.Exit
Enter your choice: 4   */
```

EXPERIMENT 2

```
/* Design, Develop and Implement a menu driven C program for the following operations on
STACK of Integers
  (Array Implementation of stack with maximum size MAX).
  a. Push an element on to stack
  b. Pop an element from stack
  c. Demonstrate Overflow and Underflow situations on stack
  d. Display the status of stack
  e. Exit
  Support the program with appropriate functions for each of the above operations.  */


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


  #define MAX 5
  int stack[MAX], top=-1;


  void push(int);
void pop();

void display();


int main()

{

    int choice=1,item;

    while(choice)

    {

        printf("\n\n........STACK OPERATIONS.........\n");

        printf("1. Push\n");

        printf("2. Pop\n");

        printf("3. Display\n");

        printf("4. Exit\n");

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

        printf("Enter your choice\n");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1: printf("Enter the element to be inserted : \t");

                     scanf("%d",&item);

                     push(item);

                     break;

            case 2: pop();

                      break;

            case 3: display();
```

```
                break;
          case 4:exit(0);
          default: printf("\n Invalid choice\n");

        }
    }
    return 0;
}

void push(int item)
{
    if(top==(MAX-1))
    {
        printf("\nSTACK OVERFLOW\n");
        return;
    }
    top=top+1;
    stack[top]=item;
}

void pop()
{
    int item;
    if(top==-1)
    {
        printf("\nSTACK UNDERFLOW\n");
        return;
    }
    item=stack[top];
    top=top-1;
    printf("The poped element is %d\t" , item);
}

void display()
{
    int i;
    if(top==-1)
    {
        printf("\n Stack is empty and nothing to display\n");
        return;
    }
    printf("\n The stack elements are: \n");
    for(i=top;i>=0;i--)
    {
        printf("%d\n",stack[i]);
    }
}

/*  OUTPUT:
```

```
........STACK OPERATIONS.........

1. Push

2. Pop

3. Display

4. Exit

.....................................

Enter your choice

1

Enter the element to be inserted :      10



........STACK OPERATIONS.........

1. Push

2. Pop

3. Display

4. Exit

.....................................

Enter your choice

1

Enter the element to be inserted :      20



........STACK OPERATIONS.........

1. Push
```

```
2. Pop

3. Display

4. Exit

.....................................

Enter your choice

1

Enter the element to be inserted :      30



........STACK OPERATIONS.........

1. Push

2. Pop

3. Display

4. Exit

.....................................

Enter your choice

1

Enter the element to be inserted :      40



........STACK OPERATIONS.........

1. Push

2. Pop

3. Display

4. Exit
```

```
....................................

Enter your choice

1

Enter the element to be inserted :      50



........STACK OPERATIONS.........

1. Push

2. Pop

3. Display

4. Exit

....................................

Enter your choice

3


 The stack elements are:

50

40

30

20

10



........STACK OPERATIONS.........

1. Push
```

```
2. Pop

3. Display

4. Exit

.....................................

Enter your choice

1

Enter the element to be inserted :      60


STACK OVERFLOW




........STACK OPERATIONS.........

1. Push

2. Pop

3. Display

4. Exit

.....................................

Enter your choice

2

The poped element is 50


........STACK OPERATIONS.........

1. Push

2. Pop

3. Display
```

```
4. Exit

.....................................

Enter your choice

2

The poped element is 40


........STACK OPERATIONS.........

1. Push

2. Pop

3. Display

4. Exit

.....................................

Enter your choice

2

The poped element is 30


........STACK OPERATIONS.........

1. Push

2. Pop

3. Display

4. Exit

.....................................

Enter your choice

2

The poped element is 20
```

```
........STACK OPERATIONS.........

1. Push

2. Pop

3. Display

4. Exit

.....................................

Enter your choice

3


 The stack elements are:

10



........STACK OPERATIONS.........

1. Push

2. Pop

3. Display

4. Exit

.....................................

Enter your choice

2

The poped element is 10


........STACK OPERATIONS.........
```

```
1. Push

2. Pop

3. Display

4. Exit

....................................

Enter your choice

2


STACK UNDERFLOW




........STACK OPERATIONS.........

1. Push

2. Pop

3. Display

4. Exit

....................................

Enter your choice

4


Process returned 0 (0x0)   execution time : 67.988 s

Press any key to continue.

*/
```

EXPERIMENT 3(A) AND 3(B)

```
/* 3a) Design , develop and implement a menu driven program in C for the following
operations on SLL of integer data.
a) Create a SLL stack of N integers
b) Display of SLL
c) Linear search
d) Concatenations of two SLL */
#include<stdio.h>
#include<stdlib.h>
struct node
{
 int data;
 struct node* link;
};
struct node * create_node()
{
 struct node* newNode;
 newNode=malloc(sizeof(struct node));
 if(newNode == NULL)
 {
 printf("No Memory allocated\n");
 return;
 }
 printf("Enter the data\n");
 scanf("%d",&newNode->data);
 newNode->link=NULL;
 return newNode;
};
struct node* insert_end(struct node * head)
{
 struct node *temp,*newNode;
 temp=head;
 newNode=create_node();
 if(head==NULL)
 {
 head=newNode;
 temp=head;
 }
 else
 {
 while(temp->link!=NULL)
 temp=temp->link;
 temp->link=newNode;
 }
 return head;
}
void display(struct node *head)
{
 struct node* temp;
 if(head==NULL)
 {
 printf("\nEmpty list\n");
 return;
 }
 temp=head;
 printf("\n The elements are : ");
 while(temp)
 {
 printf(" %d ",temp->data);
 temp=temp->link;
 }
 printf("\n");
}
void search_list(struct node* head, int key)
{
 struct node*temp=head;
 int pos=0;
 while(temp!=NULL)
 {
 pos++;
 if(key==temp->data)
 {
 printf("\nThe key element %d found at position :%d \n ",temp->data,pos);
 return;
 }
 temp=temp->link;
 }
 printf("\nThe key element %d not found in the list\n",key);
}
struct node* create_list()
{
 struct node *temp,*head=NULL,*newNode;
 int n,i;
 printf(" How many nodes? : ");
 scanf("%d", &n);
 for(i=0;i<n;i++)
 {
 newNode=create_node();
 if(head==NULL)
 head=temp=newNode;
 else
 {
 temp->link=newNode;
 temp=newNode;
 }
 }
 return head;
}
struct node* concatenate_list(struct node* head1, struct node* head2)
```

```
{
 struct node *temp;
 if(head1==NULL)
 return head2;
 if(head2==NULL)
 return head1;
 temp=head1;
 while(temp->link!=NULL)
 temp=temp->link;
 temp->link=head2;
 return head1;
}
int main()
{
 int choice,ch=1,key,n,i;
 struct node* stack=NULL, *list1,*list2,*list;
 while(1)
 {
 printf("\n");
 printf("1. Create a stack of N integers\n");
 printf("2. Display\n");
 printf("3. Linear search\n");
 printf("4. Concatenation of two SLL\n");
 printf("5. Exit\n");
 printf("Enter your choice\n");
```

```
scanf("%d",&choice);

switch(choice)

{

case 1: printf("Creating a STACK of N integers\n");

printf("Enter the value of N\n");

scanf("%d",&n);

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

stack=insert_end(stack);

break;

case 2: display(stack);

break;

case 3: printf("Enter the key element to search: \t");

scanf("%d",&key);

search_list(stack,key);

break;

case 4: printf("\nCreate List1\n");

list1=create_list();

printf("\nCreate List2\n");

list2=create_list();

printf("\nList1: \n");

display(list1);

printf("\nList2: \n");

display(list2);

list=concatenate_list(list1,list2);

printf("\n The concatenated list is : \n");
```

```
 display(list);
 break;
 case 5: exit(1);
 default: printf("Enter the correct choice\n");
 break;
 }
 }
 return 0;
}
/*output
1. Create a stack of N integers
2. Display
3. Linear search
4. Concatenation of two SLL
5. Exit
Enter your choice
1
Creating a STACK of N integers
Enter the value of N
5
Enter the data
2
Enter the data
9
Enter the data
```

```
5
Enter the data
4
Enter the data
1
1. Create a stack of N integers
2. Display
3. Linear search
4. Concatenation of two SLL
5. Exit
Enter your choice
2
The elements are : 2 9 5 4 1
1. Create a stack of N integers
2. Display
3. Linear search
4. Concatenation of two SLL
5. Exit
Enter your choice
3
Enter the key element to search: 5
The key element 5 found at position :3
1. Create a stack of N integers
2. Display
3. Linear search
```

```
4. Concatenation of two SLL

5. Exit

Enter your choice

3

Enter the key element to search: 10

The key element 10 not found in the list

1. Create a stack of N integers

2. Display

3. Linear search

4. Concatenation of two SLL

5. Exit

Enter your choice

4

Create List1

 How many nodes? : 3

Enter the data

2

Enter the data

4

Enter the data

1

Create List2

 How many nodes? : 2

Enter the data

7
```

```
Enter the data
5
List1:
The elements are : 2 4 1
List2:
The elements are : 7 5
The concatenated list is :
The elements are : 2 4 1 7 5
1. Create a stack of N integers
2. Display
3. Linear search
4. Concatenation of two SLL
5. Exit
Enter your choice
*/
/* 3b. Create a SLL Queue of N Students Data. */
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node
{
 int sem,phno;
 char name[20],branch[20],usn[20];
 struct node *link;
}*head=NULL,*newnode;
```

```
void create()
{
 int sem,phno;
 char name[20],branch[20],usn[20];
 newnode=(struct node*)malloc(sizeof(struct node));
 printf("Enter the student details\n");
 printf("USN: \n");
 scanf("%s",&usn);
 printf("Name: \n");
 scanf("%s",&name);
 printf("Branch: \n");
 scanf("%s",&branch);
 printf("Sem: \n");
 scanf("%d",&sem);
 printf("Phone Number: \n");
 scanf("%d",&phno);
 strcpy(newnode->usn,usn);
 strcpy(newnode->name,name);
 strcpy(newnode->branch,branch);
 newnode->sem=sem;
 newnode->phno=phno;
 newnode->link=NULL;
}
void insert_end()
 {
```

```
 struct node *temp;
 if(head==NULL)
 {
 create();
 head=newnode;
 temp=head;
 }
 else
 {
 create();
 temp->link=newnode;
 temp=newnode;
 }
 }
void display()
 {
 struct node* temp=head;
 if(head==NULL)
 {
 printf("\n Queue is empty");
 return;
 }
 printf("The Student details: \n");
 while(temp!=NULL)
 {
```

```
 printf("USN:%s\nName: %s\nBranch: %s\nSem: %d\nPhone Number: %d\n\n",temp-
>usn,temp-

>name,temp->branch,temp->sem,temp->phno);

 temp=temp->link;

 }

 }

void delete_front()

 {

 struct node *temp;

 temp=head;

 if(temp->link==NULL)

 {

 free(temp);

 head=NULL;

 return 0;

 }

 else

 {

 head=temp->link;

 printf("USN:%s\nName: %s\nBranch: %s\nSem: %d\nPhone Number: %d\n",temp->usn,temp-

>name,temp->branch,temp->sem,temp->phno);

 free(temp);

 }

 return 0;

 }

int main()
```

```
{
 int ch=1,n,i;
 while(ch)
 {
 printf("1. Create a SLL Queue of N Students data\n");
 printf("2. Delete Queue\n");
 printf("3. Display\n");
 printf("4. Quit\n");
 printf("enter your choice\n");
 scanf("%d",&ch);
 switch(ch)
 {
 case 1: printf("Creating Queue of N student list\n");
 printf("\nEnter the value of N \n");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 insert_end();
 break;
 case 2: delete_front();
 break;
 case 3: display();
 break;
 case 4: exit(1);
 default: printf("Wrong choice\n");
 break;
 }
 }
 return 0;
}
/* Output:
1. Create a SLL Queue of N Students data
2. Delete Queue
3. Display
4. Quit
enter your choice
1
Creating Queue of N student list
Enter the value of N
3
Enter the student details
USN:
12
Name:
abc
Branch:
cs
Sem:
3
Phone Number:
75342234
```

```
Enter the student details
USN:
32
Name:
xyz
Branch:
aiml
Sem:
4
Phone Number:
32453453
Enter the student details
USN:
45
Name:
qwert
Branch:
aiml
Sem:
3
Phone Number:
987654546
1. Create a SLL Queue of N Students data
2. Delete Queue
3. Display
```

```
4. Quit
enter your choice
3
The Student details:
USN:12
Name: abc
Branch: cs
Sem: 3
Phone Number: 75342234
USN:32
Name: xyz
Branch: aiml
Sem: 4
Phone Number: 32453453
USN:45
Name: qwert
Branch: aiml
Sem: 3
Phone Number: 987654546
1. Create a SLL Queue of N Students data
2. Delete Queue
3. Display
4. Quit
enter your choice
2
```

```
USN:12

Name: abc

Branch: cs

Sem: 3

Phone Number: 75342234

1. Create a SLL Queue of N Students data

2. Delete Queue

3. Display

4. Quit

enter your choice

3

The Student details:

USN:32

Name: xyz

Branch: aiml

Sem: 4

Phone Number: 32453453

USN:45

Name: qwert

Branch: aiml

Sem: 3

Phone Number: 987654546

1. Create a SLL Queue of N Students data

2. Delete Queue

3. Display
```

```
4. Quit

enter your choice

*/
```

EXPERIMENT 4

```
/* Design, Develop and Implement a menu driven program in C for the following operation on
Doubly linked list(DLL)
of professor data with the fieldss: ID, Name,Branch,Area of specialization
a)Create a DLL stack of N Professors Data
b) Create a DLL queue of N Professors Data
c) Display the status of DLL and count the number of nodes in it.

*/

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node
{
   struct node* prev;
   int id;
   char name[20], branch[20], area[20];
   struct node* next;
}*head=NULL,*newnode;
int count=0;
void create()
{
   int id,n,i;
   char name[20],branch[20],area[20];
   newnode=(struct node *)malloc(sizeof(struct node));
   newnode->next=NULL;
   newnode->prev=NULL;
   printf("Enter the professor details\n");
   printf("\n ID: ");
   scanf("%d",&id);
   printf("\n Name: ");
   scanf("%s",name);
   printf("\n Branch: ");
   scanf("%s",branch);
   printf("\n Area:  ");
   scanf("%s",area);
   newnode->id=id;
```

```
    strcpy(newnode->name,name);
    strcpy(newnode->branch,branch);
    strcpy(newnode->area,area);
}

void insert_end()
{
    struct node* temp;
    int n,i;
    printf("Enter the number of professors\n");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        create();
        printf("Node created");
        if(head==NULL)
        {
            printf("%d",head);
            newnode->next=head;
            head=newnode;
            head->prev=newnode;
        }
    else
        {
            temp=head;
            while(temp->next!=NULL)
                temp=temp->next;
            temp->next=newnode;
            newnode->prev=temp;

        }
    }

}

void delete_end()
{
    struct node* temp,*prevnode;
    temp=head;
    if(temp->next==NULL)
    {
        free(temp);
        head=NULL;
    }
    else
    {
        while(temp->next!=NULL)
       {
           prevnode=temp;
           temp=temp->next;
        }
        printf("The deleted data is\n");
        printf("ID: %d\nName: %s\nBranch: %s\nArea:%s\n",temp->id,temp->name,temp-
>branch,temp->area);
        free(temp);
        prevnode->next=NULL;

    }
}


void delete_front()
{
    struct node* temp;
    temp=head;
    if(temp->next==NULL)
    {
        free(temp);
        head=NULL;
    }
    else
    {
        head=temp->next;
        printf("ID: %d\nName:%s\n,Branch:%s\n,Area:%s\n",temp->id,temp->name,temp-
>branch,temp->area);
        free(temp);
    }

}

void display()
{
    struct node*temp=head;
    if(head==NULL)
    {
        printf("Empty\n");
        return;
    }
    count=0;
    printf("The professor details..\n");
    while(temp!=NULL)
    {
        printf("ID:%d\nName:%s\nBranch:%s\nArea:%s\n",temp->id,temp->name,temp-
>branch,temp->area);
        temp=temp->next;
        count++;
    }
    printf("The number of nodes:  %d\n",count);

}

int main()
{
    int ch=1,op=1,n,i;
    while(ch)
    {
        printf(" 1. create a DLL stack of N professors data\n");
        printf(" 2. create a DLL queue of N professors data\n");
        printf(" 3. Display the status with number of nodes in it\n");
        printf(" 4. exit\n");
        printf("Enter your choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: while(op)
                    {
                    printf("Stack of N professors\n");
                    printf("1.push the data\n");
                    printf("2.pop the data\n");
                    printf(" Enter 0 to exit\n");
                    printf("Enter your option\n");
                    scanf("%d",&op);
                    switch(op)
                    {
                        case 1:insert_end();
                                break;
                        case 2: delete_end();
                                break;

                        default: printf("Enter the correct choice\n");
                                break;
                    }
                    }
                    op=1;
                    break;
                case 2: while(op)
                        {
                            printf("Queue of professors\n");
                            printf("1. Insert to the queue\n");
                            printf("2. Delete from the queue\n");
                            printf("Enter 0 to exit\n");
                            printf("Enter your choice\n");
                            scanf("%d",&op);
                            switch(op)
                            {
                                case 1: insert_end();
                                    break;
                                case 2: delete_front();
                                    break;

                                default: printf("Enter the correct choice\n");
                                     break;
                            }
                        }
                        op=1;
                        break;
                case 3: printf("Displaying...\n");
                        display();
                        break;
                case 4: exit(1);
                default: printf("Enter the correct choice\n");
                     break;

            }
        }
return 0;
}
```

EXPERIMENT 5

LAB PROGRAM 5
/* Given an array of elements, construct a complete binary tree from this array in level order fashion.
that is elements from left in the array will be filled in the tree level wise starting from level 0.
Ex: Input arr[]={1,2,3,4,5,6}

```
    output: Root of the following tree
     1
    / \
   2    3
  /\    /\
 4  5  6
 */

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

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

struct Node* newNode(int data)
{
	struct Node* node = (struct Node*)malloc(sizeof(struct Node));
	node->data = data;
	node->left = node->right = NULL;
	return (node);
}


struct Node* insertLevelOrder(int arr[],int i, int n)
{
	struct Node *root =NULL;

	if (i < n)
	{
		root = newNode(arr[i]);
		root->left = insertLevelOrder(arr,2 * i + 1, n);
		root->right = insertLevelOrder(arr,2 * i + 2, n);
	}
	return root;
}


void inOrder(struct Node* root)
{
	if (root != NULL)
	{
		inOrder(root->left);
		printf("%d  ",root->data);
		inOrder(root->right);
	}
}

int main()
{
	int arr[20],n,i;
	printf("Enter the number of nodes\n");
	scanf("%d",&n);
	printf("Enter the tree node elements\n");
	for(i=0;i<n;i++)
```

```
    scanf("%d",&arr[i]);
        struct Node* root = insertLevelOrder(arr, 0, n);
        inOrder(root);

}

/*OUTPUT

Enter the number of nodes
6
Enter the tree node elements
1
2
3
4
5
6
4  2  5  1  6  3
Process returned 0 (0x0)   execution time : 5.882 s
Press any key to continue.
*/
```

EXPERIMENT 6

```
/* Design, Develop and Implement a menu driven Program in C for the following operations on
Binary Search Tree (BST) of Integers
        a. Create a BST of N Integers
         b. Traverse the BST in Inorder, Preorder and Post Order
        c. Search the BST for a given element (KEY) and report the appropriate message */

#include <stdio.h>
#include <stdlib.h>
int flag=0;
struct BST
{
   int data;
   struct BST *left,*right;
};

/*FUNCTION PROTOTYPE*/
void insert(struct BST *, struct BST *);
void inorder(struct BST *);
void preorder(struct BST *);
void postorder(struct BST *);
struct BST *search(struct BST *, int, struct BST **);
```

```
int main()
{
   int choice;
   int ans =1;
   int key;
   struct BST *newnode, *root, *tmp, *parent;
   struct BST *get_node();
   root = NULL;
   printf("\nProgram For Binary Search Tree ");
   do
   {
      printf("\n 1.Create");
      printf("\n 2.Search");
      printf("\n 3.Recursive Traversals");
      printf("\n 4.Exit");
      printf("\n Enter your choice :");
      scanf("%d", &choice);
      switch (choice)
      {
         case 1:
               do
               {
                  newnode = get_node();
                  printf("\n Enter The Element ");
                  scanf("%d", &newnode->data);
                  if (root == NULL) /* Tree is not Created */
                     root = newnode;
                  else
                     insert(root, newnode);
                  printf("\n Want To enter More Elements?(1/0)");
                  scanf("%d",&ans);
               } while (ans);
               break;
         case 2:
               printf("\n Enter Element to be searched :");
               scanf("%d", &key);
               tmp = search(root, key, &parent);
               if(flag==1)
               {
                  printf("\n Parent of node %d is %d", tmp->data, parent->data);
               }
               else
               {
                  printf("\n The %d Element is not Present",key);
               }
               flag=0;
```

```
                break;
            case 3:
                if (root == NULL)
                printf("Tree Is Not Created");
                else
                {
                   printf("\nThe Inor\der display :\n");
                   inorder(root);
                   printf("\nThe Preorder display : \n");
                   preorder(root);
                   printf("\nThe Postorder display : \n");
                   postorder(root);
                }
                break;
            }
    }
    while (choice != 4);
 }

/*Get new Node */
struct BST *get_node()
{
    struct BST *temp;
    temp = (struct BST *) malloc(sizeof(struct BST));
    temp->left = NULL;
    temp->right = NULL;
    return temp;
}

/*This function is for creating a binary search tree */


void insert(struct BST *root, struct BST *newnode)
{
    if (newnode->data < root->data)
    {
        if(root->left==NULL)
            root->left=newnode;
        else
            insert(root->left, newnode);
    }
    if (newnode->data > root->data)
    {
        if (root->right == NULL)
            root->right = newnode;
        else
```

```
            insert(root->right, newnode);
    }
}
/*This function is for searching the node from binary Search Tree*/
struct BST *search(struct BST *root, int key, struct BST **parent)
{
    struct BST * temp;
    temp = root;
    while (temp != NULL)
    {
        if (temp->data == key)
        {
            printf("\nThe %d Element is Present", temp->data);
            flag=1;
            return temp;
        }
        *parent = temp;
        if (temp->data > key)
            temp = temp->left;
        else
            temp = temp->right;
    }
    return NULL;
}

/*This function displays the tree in inorder fashion */
void inorder(struct BST *temp)
{
    if (temp != NULL)
    {
        inorder(temp->left);
        printf("%d\t", temp->data);
        inorder(temp->right);
    }
}
/*This function displays the tree in preorder fashion */
void preorder(struct BST *temp)
{
    if (temp != NULL)
    {
        printf("%d\t", temp->data);
        preorder(temp->left);
        preorder(temp->right);
    }
}
/*This function displays the tree in postorder fashion */
```

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

EXPERIMENT 7

```
/* Design , develop and implement a program in C for the following operations on graph G of
cities
 a. create a graph of N cities using adjacency matrix
 b. print all the nodes reachable from a given starting node in a digraph using DFS/BFS method.
*/

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

int a[20][20],q[20],visited[20],reach[10],n,i,j,f=0,r= -1,count=0;

void bfs(int v)
{
    for(i=1;i<=n;i++)
        if(a[v][i] && !visited[i])
            q[++r]=i;
    if(f<=r)
    {
        visited[q[f]]=1;
        bfs(q[f++]);
    }
}

void dfs(int v)
{
    int i;
    reach[v]=1;
    for(i=1;i<=n;i++)
    {
        if(a[v][i] && !reach[i])
        {
            printf("\n %d->%d",v,i);
            count++;
            dfs(i);
        }
    }
}

void main()
{
    int v, choice;
    while(1)
    {
        printf("1.Create a graph of N cities\n");
        printf("2.BFS\n");
        printf("3.DFS\n");
        printf("4.Exit\n");
        printf("Enter your choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("\n Enter the number of cities:");
                    scanf("%d",&n);
                    for(i=1;i<=n;i++)
                    {
                        q[i]=0;
                        visited[i]=0;
                    }
                    for(i=1;i<=n-1;i++)
                        reach[i]=0;
                    printf("\n Enter graph data in adjacency matrix form:\n");
                    for(i=1;i<=n;i++)
                        for(j=1;j<=n;j++)
                            scanf("%d",&a[i][j]);
                    break;
            case 2: printf("Print using BFS...\n");
                    printf("\n Enter the starting vertex:");
                    scanf("%d",&v);
                    bfs(v);
                    if((v<1)||(v>n))
                    {
                        printf("\n Bfs is not possible");
                    }
                    else
                    {
                        printf("\n The nodes which are reachable from %d:\n",v);
                        for(i=1;i<=n;i++)
                            if(visited[i])
                                printf("%d\t",i);
                    }
                    break;
            case 3: printf("Print using DFS and check for connected or not connected\n");
                    dfs(1);
                    if(count==n-1)
                        printf("\n Graph is connected\n");
                    else
                        printf("\n Graph is not connected\n");
                    break;
            case 4: exit(0);
            default: printf("Enter correct choice\n");
                    break;
        }

    }
}
```

EXPERIMENT 8

```
/*Design and develop a program in C that uses hash function H:k->L as H(K)=K mod m (
reminder method)
and implement hashing technique to map a given key K to the address space L. Resolve the
collission (if any)
using linear probing. */

#include <stdio.h>
#include <stdlib.h>
#define MAX 10

int create(int num)
  {
    int key;
    key=num%100;
    return key;
  }

void linear_prob(int a[MAX], int key, int num)
  {
    int flag, i, count=0;
    flag=0;
    if(a[key]== -1)
```

```
        {
           a[key] = num;
        }
        else
        {
           printf("\nCollision Detected...!!!\n");
           i=0;
           while(i<MAX)
           {
              if (a[i]!=-1)
              count++;
              i++;
           }
           printf("Collision avoided successfully using LINEAR PROBING\n");
           if(count == MAX)
           {
              printf("\n Hash table is full");
              display(a);
              exit(1);
           }
           for(i=key+1; i<MAX; i++)
              if(a[i] == -1)
              {
                 a[i] = num;
                 flag =1;
                 break;
              }
           i=0;
           while((i<key) && (flag==0))
           {
              if(a[i] == -1)
              {
                 a[i] = num;
                 flag=1;
                 break;
              }
              i++;
           }
        }
     }

  void display(int a[MAX])
     {
        int i;
        printf("\n\n Displaying all: The hash table is\n");
        for(i=0; i<MAX; i++)
```

```
            printf("\n %d %d ", i, a[i]);
        printf("\n Filtered Display:  The hash table is\n\n");
        for(i=0; i<MAX; i++)
            if(a[i]!=-1)
            {
                printf("\n %d %d ", i, a[i]);
                continue;
            }
    }

void main()
    {
        int a[MAX],num,key,i;
        int ans=1;
        printf(" Hashing : Collision handling by linear probing : \n");
        for (i=0;i<MAX;i++)
        {
            a[i] = -1;
        }
        do
        {
            printf("\n Enter the data");
            scanf("%4d", &num);
            key=create(num);
            linear_prob(a,key,num);
            printf("\n Do you wish to continue ? (1/0) ");
            scanf("%d",&ans);
        }while(ans);
        display(a);
    }
```