Week 1

AL Lab

Aniket sambher

Section A

Roll no-58

Reg no-190905466

1). Write a program to construct a binary tree to support the following operations. Assume no duplicate elements while constructing the tree.
i. Given a key, perform a search in the binary search tree. If the key is found then display “key found” else insert the key in the binary search tree.
ii. Display the tree using inorder, preorder and post order traversal methods

```
#include<stdio.h>

#include<stdlib.h>

typedef struct node* link;

typedef struct node {

int data;

link right;

link left;


}node;

void inorder(link root){

        if(root==NULL)

                return ;

        inorder(root->left);

        printf("%d\t",root->data);

        inorder(root->right);

}

void preorder(link root){


        if(root==NULL)

                return ;
```

```
		printf("%d\t",root->data);
		preorder(root->left);
		preorder(root->right);
}
void postorder(link root){
		if(root==NULL)
			return ;
		postorder(root->left);
		postorder(root->right);
		printf("%d\t",root->data);
}
link createnode(int data){
		link temp = malloc(sizeof(node));
		temp->left = temp -> right = NULL;
		temp->data = data;
		return temp;
}
link insert(link root,int data){
		if(root==NULL){
			return createnode(data);
		}
		else if(data<root->data)
			root->left=insert(root->left,data);
		else if(data>root->data)
			root->right=insert(root->right,data);
		else{
			printf("duplicate item \n");


		}
		return root;
```

```
}
int search(link root,int ele){
	if(root==NULL)
		return 0;
	if(root->data==ele)
		return 1;
	else if(root->data>ele)
		return search(root->left,ele);
	else
		return search(root->right,ele);


}


void main(){
	link root = NULL;

	int ch = 1;
	int x;
	printf("1. Insert    2. Inorder    3. Preorder    4. Postorder  5.Search  0. Exit\n");

	while(ch)
	{
		printf("\nCommand: ");
		scanf("%d", &ch);

		switch(ch)
		{case 1:printf("enter the number");
```

```
                scanf("%d",&x);
                root=insert(root,x);
                break;
        case 2:printf("inorder");
                inorder(root);
                break;
        case 3:printf("preorder");
                preorder(root);
                break;
        case 4:printf("Postorder");
                postorder(root);
                break;
        case 5:printf("Search\n");
                printf("enter the number to be searched\n");
                scanf("%d",&x);
                if(search(root,x))
                        printf("found");
                else{
                        printf("not found inserting in bst");
                        root=insert(root,x);
                }
                break;
        case 0:printf("Exit");

                break;

        }

}
}
```

```
Command: 1
enter the number5

Command: 1
enter the number3

Command: 1
enter the number7

Command: 2
inorder3        5       7
Command: 3
preorder5       3       7
Command: 4
Postorder3      7       5
Command: 5
Search
enter the number to be searched
3
found
Command: 5
Search
enter the number to be searched
9
not found inserting in bst
Command:
```

2). Write a program to implement the following graph representations and display them.
i. Adjacency list
ii. Adjacency matrix

```
#include <stdio.h>

#include <stdlib.h>

typedef struct node* link;

typedef struct node

{

   int data;

   link next;

} node;

link createNode(int x)

{
```

```
    link temp = (link)malloc(sizeof(node));
    temp -> data = x;
    temp -> next = NULL;
    return temp;
}
link insertRear(link head, int ele)
{
    if (head == NULL)
    {
        head = createNode(ele);
        return head;
    }
    link rear = head;
    while (rear -> next != NULL)
        rear = rear -> next;
    link temp = createNode(ele);
    rear -> next = temp;
    return head;
}
void display(link head)
{
    link temp = head;
    printf("%d  ", temp->data);
    temp = temp -> next;
    while (temp != NULL)
    {
        printf("%d  ", temp -> data);
        temp = temp -> next;
    }

}
```

```
int main()
{
   int e, v;
   printf("enter the no of vertex: ");
   scanf("%d", &v);
   link* list = malloc(v * sizeof(link));
   for(int i = 0 ; i < v ; i++)
      list[i] = insertRear(list[i], i);
   int **mat = malloc(v * sizeof(int*));
   for(int i = 0 ; i < v ; i++)
      mat[i] = malloc(v * sizeof(int));
   for(int i = 0 ; i < v ; i++)
      for(int j = 0 ; j < v ; j++)
         mat[i][j] = 0;
   printf("enter the  no of edges ");
   scanf("%d", &e);
   for(int i = 0 ; i < e ; i++)
   {
      printf("enter starting vertex and terminating vertex of edge %d\n", i + 1);
      int a, b;
      scanf("%d %d", &a, &b);
      mat[a][b]=1;
      list[a] = insertRear(list[a], b);
}
   printf("\nAdjacency List\n");
   for(int i = 0 ; i < v ; i++)
   {
   display(list[i]);
   printf("\n");
   }
   printf("\n\nAdjacency matrix\n");
```

```
for(int i = 0 ; i < v ; i++)
{
   for(int j = 0 ; j < v ; j++)
      printf("%d\t", mat[i][j]);
      printf("\n");
}
}
```

```
enter the no of vertex: 3
enter the  no of edges 3
enter starting vertex and terminating vertex of edge 1
0 1
enter starting vertex and terminating vertex of edge 2
1 2
enter starting vertex and terminating vertex of edge 3
2 0

Adjacency List
0  1
1  2
2  0


Adjacency matrix
0       1       0
0       0       1
1       0       0


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