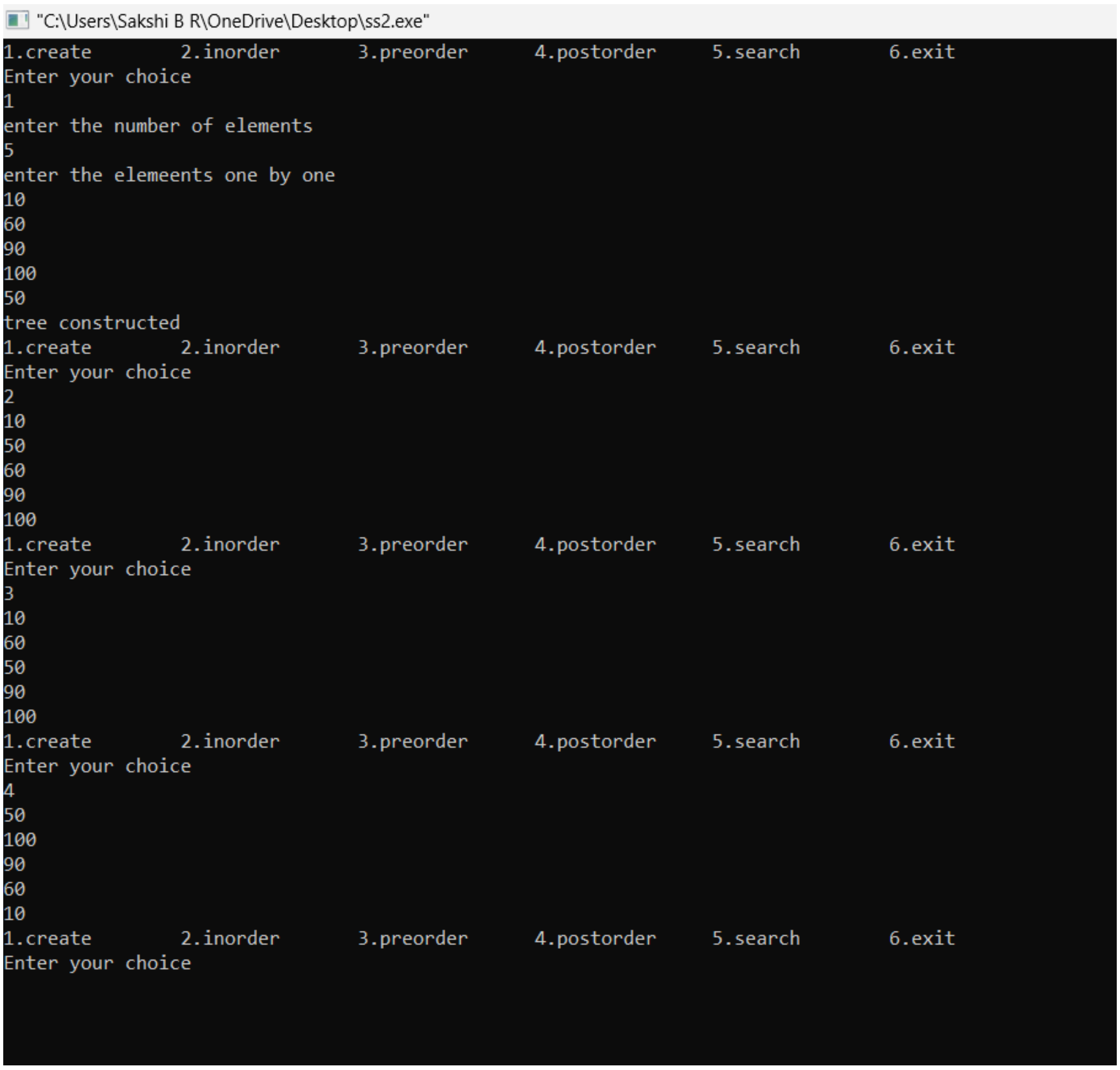
Lab 8

```
#include<stdio.h>
#include<stdlib.h>
typedef struct NODE
{
    int info;
    struct NODE *lchild;
    struct NODE *rchild;
}NODE;
NODE *root=NULL;
void create();
void insert(int);
void inorder(NODE *);
void preorder(NODE *);
void postorder(NODE *);
void search(NODE *,int);
int main()
{
    int ch,key;
    do
    {
        printf("1.create\t2.inorder\t3.preorder\t4.postorder\t5.search\t6.exit\n");
        printf("Enter your choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1 : create();
            break;
            case 2 : inorder(root);
            break;
            case 3 : preorder(root);
            break;
            case 4 : postorder(root);
            break;
            case 5 : printf("enter the key\n");
            scanf("%d",&key);
            search(root,key);
            break;
            case 6 : exit(0);
            default : printf("Invalid choice");
        }
    }while(ch!=6);
    return 0;
}
void create()
{
    int n,i,e;
    printf("enter the number of elements\n");
    scanf("%d",&n);
    printf("enter the elemeents one by one\n");
    for(i=1;i<=n;i++)
    {
        scanf("%d",&e);
        insert(e);
    }
    printf("tree constructed\n");
}
void insert(int e)
{
    NODE *nn,*temp,*prev;
    nn=(NODE *)malloc(sizeof(NODE));
    nn->info=e;
    nn->lchild=NULL;
    nn->rchild=NULL;
    if(root==NULL)
    {
        root=nn;
        return;
    }
    temp=root;
    while(temp!=NULL)
    {
        prev=temp;
        if(e<temp->info)
        temp=temp->lchild;
        else if(e>temp->info)
        temp=temp->rchild;
        else
        {
            printf("its a duplicate node");
            return;
        }
    }
    if(e<prev->info)
    prev->lchild=nn;
    else
    prev->rchild=nn;
}
void inorder(NODE *tree)
{
    if(tree!=NULL)
    {
        inorder(tree->lchild);
        printf("%d\n",tree->info);
        inorder(tree->rchild);
    }
}
void preorder(NODE *tree)
{
    if(tree!=NULL){
    printf("%d\n",tree->info);
    preorder(tree->lchild);
    preorder(tree->rchild);
    }
}
void postorder(NODE *tree)
{
    if(tree!=NULL)
    {
        postorder(tree->lchild);
        postorder(tree->rchild);
        printf("%d\n",tree->info);
    }
}
void search(NODE *tree,int key)
{
    if(tree==NULL)
    {
        printf("key not found\n");
        return;
    }
    else if(tree->info==key)
    {
        printf("key found\n");
        return;
    }
    else if(key<tree->info)
    search(tree->lchild,key);
    else
    search(tree->rchild,key);
}
```



```
"C:\Users\Sakshi B R\OneDrive\Desktop\ss2.exe"
1.create        2.inorder       3.preorder      4.postorder     5.search        6.exit
Enter your choice
1
enter the number of elements
5
enter the elemeents one by one
10
60
90
100
50
tree constructed
1.create        2.inorder       3.preorder      4.postorder     5.search        6.exit
Enter your choice
2
10
50
60
90
100
1.create        2.inorder       3.preorder      4.postorder     5.search        6.exit
Enter your choice
3
10
60
50
90
100
1.create        2.inorder       3.preorder      4.postorder     5.search        6.exit
Enter your choice
4
50
100
90
60
10
1.create        2.inorder       3.preorder      4.postorder     5.search        6.exit
Enter your choice
```