```
/*************************************************************************************
*
Subject: DSA Laboratory
Practical No: 08
Title: Write a C++ Program to create an Optimal BST.
   Input:
        (Key,Frequencies) = ((10,8), (20,4) (30,6))
   Output:
        a) Display (Keys & Frequencies)
        b) Sorted Keys as per their Frequencies
        c) Create and Display OBST.
*************************************************************************************
*/
                //…….Header Files
#include <iostream>
using namespace std;

                //…….Number of Keys

int Keys = 3;

                //…….Class to create an Array of Objects of (key, freq)

class OBST
{
   int key;
   int freq;
   public:
      void init_Keys();
      void bubble_Sort();
      void create_OBST();
      void show_Preorder(struct BSTNode *root);
}obj[3];

                //…….Member Function to Accept and Display Object's (key, freq)

void OBST::init_Keys()
{
   int i;

   for(i=0; i<Keys; i++)
   {
      cout<<"\n\t Enter Key: ";
      cin>>obj[i].key;

      cout<<"\n\t Enter Frequency: ";
```

```
        cin>>obj[i].freq;
    }

    cout<<"\n\t Key Frequency";
    for(i=0; i<Keys; i++)
    {
        cout<<"\n\t  "<<obj[i].key;

        cout<<"\t"<<obj[i].freq;
    }
}
```

//……Member Function to Sort Keys in Decreasing Order of their Frequencies

```
void OBST::bubble_Sort()
{
    int i,j;
    int Key;
    int Freq;

    for(i=0; i<Keys-1; i++)
    {
        for(j=0; j<Keys-1; j++)
        {
            if(obj[j].freq < obj[j+1].freq)
            {
                Key = obj[j].key;
                Freq = obj[j].freq;

                obj[j].key = obj[j+1].key;
                obj[j].freq = obj[j+1].freq;

                obj[j+1].key = Key;
                obj[j+1].freq = Freq;
            }
        }
    }

    cout<<"\n\t Key Frequency";
    for(i=0; i<Keys; i++)
    {
        cout<<"\n\t  "<<obj[i].key;

        cout<<"\t"<<obj[i].freq;
    }
}
```

//…….Structure of Node of BST

```
struct BSTNode
{
   int key;
   int freq;
   struct BSTNode *left;
   struct BSTNode *right;
}*Root;
```

//…….Member Function to create an OBST

```
void OBST::create_OBST()
{
   int i;
   int done;
   struct BSTNode *Newnode, *current;

   i = 0;
   while(i < Keys)
   {
      done = 0;

      Newnode = new struct BSTNode;

      Newnode->key = obj[i].key;
      Newnode->freq = obj[i].freq;
      Newnode->left = NULL;
      Newnode->right = NULL;

      if(Root == NULL)
      {
         Root = Newnode;
      }
      else
      {
      current = Root;
      while(!done)
      {
         if(Newnode->key < current->key)
         {
            if(current->left == NULL)
            {
               current->left = Newnode;
               done = 1;
```

```
            }
            else
                current = current->left;
        }
        else
        {
            if(current->right == NULL)
            {
                current->right = Newnode;
                done = 1;
            }
            else
                current = current->right;
        }
    }
    }
    i++;
  }
}
```

//…….Member Function to display OBST in Preorder

```
int level = 1;
int cost = 0;
void OBST::show_Preorder(struct BSTNode *root)
{
  if(root)
  {
    cout<<" "<<root->key;
    cost = cost + level*root->freq;
    level++;
    this->show_Preorder(root->left);
    this->show_Preorder(root->right);
  }
}
```

//…….Main Function

```
int main()
{
  cout<<"\n -----*** A C++ Program to create an Optimal BST ***------ \n";

  OBST Obj1;

  cout<<"\n 1. Accept and Display Keys and Frequencies.....\n";
```

```
    Obj1.init_Keys();

    cout<<"\n 2. Sort Keys as per Frequencies.....\n";
    Obj1.bubble_Sort();

    cout<<"\n 3. Create OBST and Display OBST in Preorder.....\n";
    Obj1.create_OBST();

    cout<<"\n Preorder(OBST): ";
    Obj1.show_Preorder(Root);
    cout<<"\n Total Searching Cost: "<<cost;
    return 0;
}
```

/*----------**OUTPUT**----------

-----*** A C++ Program to create an Optimal BST ***------

1. Accept and Display Keys and Frequencies.....

Enter Key: 10

Enter Frequency: 8

Enter Key: 20

Enter Frequency: 4

Enter Key: 30

Enter Frequency: 6

Key Frequency
10 8
20 4
30 6

2. Sort Keys as per Frequencies.....

Key Frequency
10 8
30 6
20 4

3. Create OBST and Display OBST in Preorder.....

```
 Preorder(OBST):  10 30 20
 Total Searching Cost: 32

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

*/
```