```
/*
 * C++ Program to Implement Skip List
 */
#include <iostream>
#include <cstdlib>
#include <cmath>
#include <cstring>
#define MAX_LEVEL 6
const float P = 0.5;
using namespace std;
/*
 * Skip Node Declaration
 */
struct snode
{
    int value;
    snode **forw;
    snode(int level, int &value)
    {
        forw = new snode * [level + 1];
        memset(forw, 0, sizeof(snode*) * (level + 1));
        this->value = value;
    }
    ~snode()
    {
        delete [] forw;
    }
};
/*
 * Skip List Declaration
 */
struct skiplist
{
    snode *header;
    int value;
    int level;
    skiplist()
    {
        header = new snode(MAX_LEVEL, value);
        level = 0;
    }
    ~skiplist()
    {
        delete header;
```

```
    }
    void display();
    bool contains(int &);
    void insert_element(int &);
    void delete_element(int &);
};
/*
 * Main: Contains Menu
 */
int main()
{
    skiplist ss;
    int choice, n;
    while (1)
    {
        cout<<endl<<"-----------------------"<<endl;
        cout<<endl<<"Operations on Skip list"<<endl;
        cout<<endl<<"-----------------------"<<endl;
        cout<<"1.Insert Element"<<endl;
        cout<<"2.Delete Element"<<endl;
        cout<<"3.Search Element"<<endl;
        cout<<"4.Display List "<<endl;
        cout<<"5.Exit "<<endl;
        cout<<"Enter your choice : ";
        cin>>choice;
        switch(choice)
        {
        case 1:
             cout<<"Enter the element to be inserted: ";
             cin>>n;
             ss.insert_element(n);
             if(ss.contains(n))
                 cout<<"Element Inserted"<<endl;
             break;
        case 2:
             cout<<"Enter the element to be deleted: ";
             cin>>n;
             if(!ss.contains(n))
             {
                 cout<<"Element not found"<<endl;
                break;
             }
             ss.delete_element(n);
```

```
            if(!ss.contains(n))
                cout<<"Element Deleted"<<endl;
            break;
        case 3:
            cout<<"Enter the element to be searched: ";
            cin>>n;
            if(ss.contains(n))
                cout<<"Element "<<n<<" is in the list"<<endl;
            else
                cout<<"Element not found"<<endl;
        case 4:
            cout<<"The List is: ";
            ss.display();
            break;
        case 5:
            exit(1);
            break;
        default:
            cout<<"Wrong Choice"<<endl;
        }
    }
    return 0;
}

/*
 * Random Value Generator
 */
float frand()
{
    return (float) rand() / RAND_MAX;
}

/*
 * Random Level Generator
 */
int random_level()
{
    static bool first = true;
    if (first)
    {
        srand((unsigned)time(NULL));
        first = false;
    }
```

```
    int lvl = (int)(log(frand()) / log(1.-P));
    return lvl < MAX_LEVEL ? lvl : MAX_LEVEL;
}

/*
 * Insert Element in Skip List
 */
void skiplist::insert_element(int &value)
{
    snode *x = header;
    snode *update[MAX_LEVEL + 1];
    memset(update, 0, sizeof(snode*) * (MAX_LEVEL + 1));
    for (int i = level;i >= 0;i--)
    {
        while (x->forw[i] != NULL && x->forw[i]->value < value)
        {
            x = x->forw[i];
        }
        update[i] = x;
    }
    x = x->forw[0];
    if (x == NULL || x->value != value)
    {
        int lvl = random_level();
        if (lvl > level)
        {
            for (int i = level + 1;i <= lvl;i++)
            {
                update[i] = header;
            }
            level = lvl;
        }
        x = new snode(lvl, value);
        for (int i = 0;i <= lvl;i++)
        {
            x->forw[i] = update[i]->forw[i];
            update[i]->forw[i] = x;
        }
    }
}

/*
 * Delete Element from Skip List
```

```
 */
void skiplist::delete_element(int &value)
{
    snode *x = header;
    snode *update[MAX_LEVEL + 1];
    memset (update, 0, sizeof(snode*) * (MAX_LEVEL + 1));
    for (int i = level;i >= 0;i--)
    {
        while (x->forw[i] != NULL && x->forw[i]->value < value)
        {
            x = x->forw[i];
        }
        update[i] = x;
    }
    x = x->forw[0];
    if (x->value == value)
    {
        for (int i = 0;i <= level;i++)
        {
            if (update[i]->forw[i] != x)
                break;
            update[i]->forw[i] = x->forw[i];
        }
        delete x;
        while (level > 0 && header->forw[level] == NULL)
        {
            level--;
        }
    }
}

/*
 * Display Elements of Skip List
 */
void skiplist::display()
{
    const snode *x = header->forw[0];
    while (x != NULL)
    {
        cout << x->value;
        x = x->forw[0];
        if (x != NULL)
            cout << " - ";
```

```
    }
    cout <<endl;
}

/*
 * Search Elemets in Skip List
 */
bool skiplist::contains(int &s_value)
{
    snode *x = header;
    for (int i = level;i >= 0;i--)
    {
        while (x->forw[i] != NULL && x->forw[i]->value < s_value)
        {
            x = x->forw[i];
        }
    }
    x = x->forw[0];
    return x != NULL && x->value == s_value;
}
```

Output:

```
-----------------------

Operations on Skip list

-----------------------
1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice : 1
Enter the element to be inserted: 2
Element Inserted
```

```
----------------------
1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice : 1
Enter the element to be inserted: 4
Element Inserted
```

```
----------------------
1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice : 4
The List is: 2 - 4
```