JARIWALA SAHIL Y.

ROLL NO: CE057

SUBJECT: AA

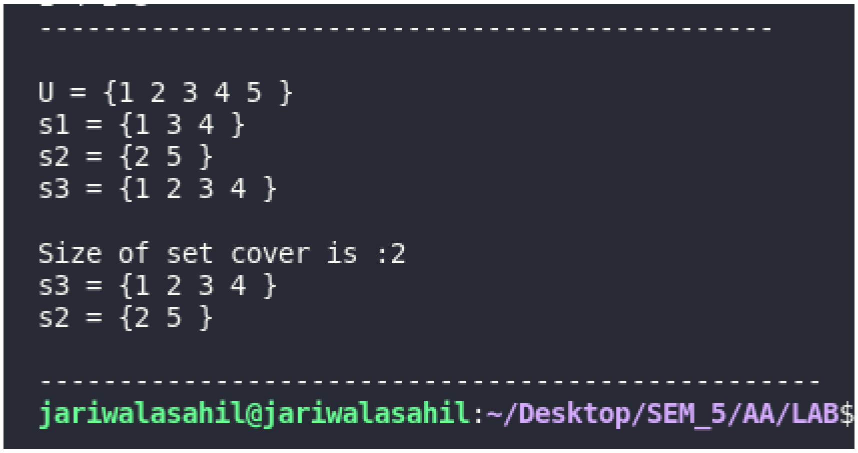
LAB : 10

AIM: Implementation of Greedy Approximation of Set Cover problem.

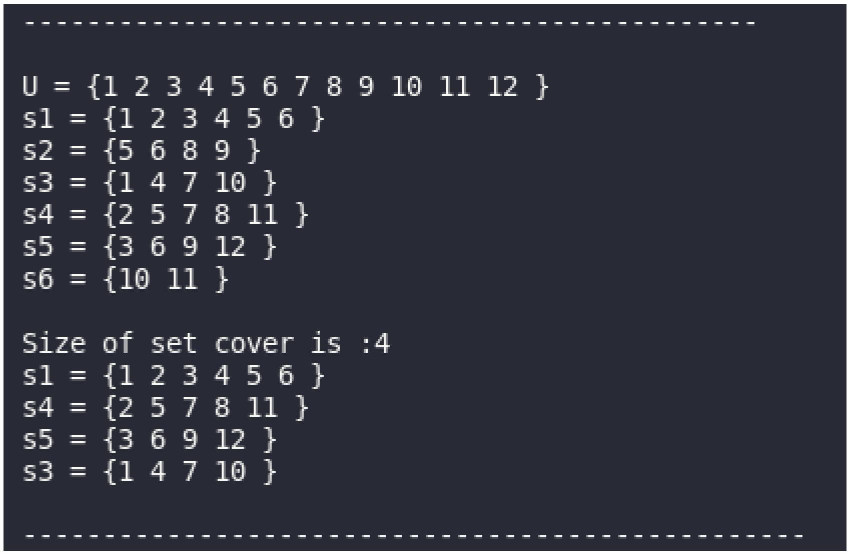
CODE:

```
#include <bits/stdc++.h>

using namespace std;

int setIntersection(set<int> s1, set<int> s2)
{
    std::vector<int> v_intersection;
    vector<int> v1(s1.begin(), s1.end());
    vector<int> v2(s2.begin(), s2.end());
    std::set_intersection(v1.begin(), v1.end(),
                           v2.begin(), v2.end(),
                           std::back_inserter(v_intersection));
    return v_intersection.size();
}

vector<int> setUnion(set<int> s1, set<int> s2)
{
    vector<int> v_union;
    vector<int> v1(s1.begin(), s1.end());
    vector<int> v2(s2.begin(), s2.end());
    set_union(v1.begin(), v1.end(),
               v2.begin(), v2.end(),
               back_inserter(v_union));
    return v_union;
}

set<int> setDiff(set<int> s1, set<int> s2)
{
    std::vector<int> v_diff;
    vector<int> v1(s1.begin(), s1.end());
    vector<int> v2(s2.begin(), s2.end());
    std::set_difference(v1.begin(), v1.end(),
                         v2.begin(), v2.end(),
                         std::back_inserter(v_diff));
    set<int> s(v_diff.begin(), v_diff.end());
    return s;
}

void setCover(set<int> universal, vector<set<int>> subsets)
{
    set<int> U = universal;
    vector<set<int>> ans;
    vector<int> idx;
    while (!U.empty())
    {
        int index = 0;
        int max = INT_MIN;
        for (int i = 0; i < subsets.size(); i++)
        {
            int chosen_set = setIntersection(U, subsets[i]);
            if (max < chosen_set)
            {
                max = chosen_set;
                index = i;
            }
        }

        idx.push_back(index);
        U = setDiff(U, subsets[index]);
        ans.push_back(subsets[index]);
    }
    cout << endl
         << "Size of set cover is :" << ans.size() << endl;
    for (int i = 0; i < ans.size(); i++)
    {
        cout << "s" << idx[i] + 1 << " = {";
        for (auto x : ans[i])
            cout << x << " ";
        cout << "}" << endl;
    }
}

int main()
{
    int u_size, no_sets, size_set;
    cout << "Enter the size of universal set:" << endl;
    cin >> u_size;
    set<int> universal;
    cout << "Enter element of universal set:" << endl;
    int element;
    for (int i = 0; i < u_size; i++)
    {
        cin >> element;
        universal.insert(element);
    }
    cout << "Enter the number of subsets:" << endl;
    cin >> no_sets;
    vector<set<int>> subsets;
    for (int i = 0; i < no_sets; i++)
    {
        cout << "Enter the size of " << i << "th set: " << endl;
        cin >> size_set;
        set<int> s;
        cout << "Enter the elements of " << i << "th set: " << endl;
        for (int j = 0; j < size_set; j++)
        {
            cin >> element;
            s.insert(element);
        }
        subsets.push_back(s);
    }
    cout << "--------------------------------------------" << endl;
    cout << endl
         << "U = {";
    for (auto j = universal.begin(); j != universal.end(); j++)
    {
        cout << *j << " ";
    }
    cout << "}" << endl;
    int k = 1;
    for (int i = 0; i < subsets.size(); i++)
    {
        cout << "s" << k << " = {";
        k++;
        for (auto x : subsets[i])
            cout << x << " ";
        cout << "}" << endl;
    }
    setCover(universal, subsets);
    cout << endl
         << "-----------------------------------------------" << endl;
}
```

OUTPUT:

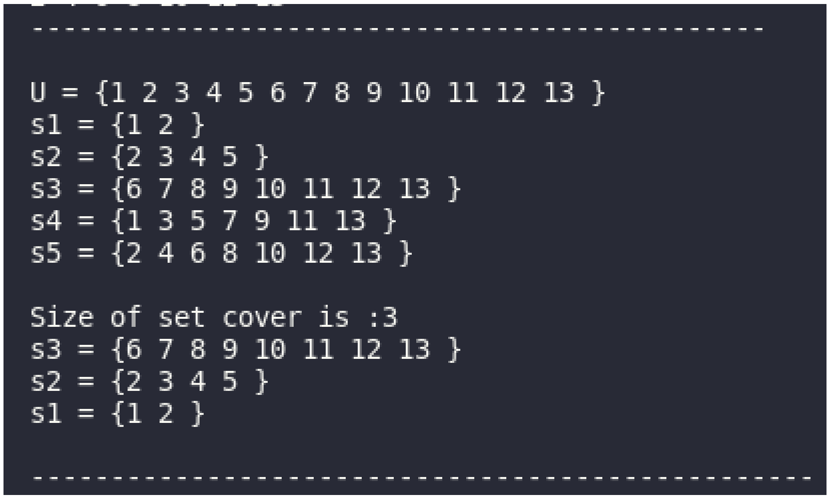


```
----------------------------------------------

U = {1 2 3 4 5 }
s1 = {1 3 4 }
s2 = {2 5 }
s3 = {1 2 3 4 }

Size of set cover is :2
s3 = {1 2 3 4 }
s2 = {2 5 }

-------------------------------------------------
jariwalasahil@jariwalasahil:~/Desktop/SEM_5/AA/LAB$
```



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

U = {1 2 3 4 5 6 7 8 9 10 11 12 }
s1 = {1 2 3 4 5 6 }
s2 = {5 6 8 9 }
s3 = {1 4 7 10 }
s4 = {2 5 7 8 11 }
s5 = {3 6 9 12 }
s6 = {10 11 }

Size of set cover is :4
s1 = {1 2 3 4 5 6 }
s4 = {2 5 7 8 11 }
s5 = {3 6 9 12 }
s3 = {1 4 7 10 }

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

IN THIS CASE GREEDY APPROCH FAILS IT WILL NOT GIVE OPTIMAL ANS HERE ITS GIVE SET COVER AS 3 BUT OPTIMAL IS 2 FOR S4 AND S5



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

U = {1 2 3 4 5 6 7 8 9 10 11 12 13 }
s1 = {1 2 }
s2 = {2 3 4 5 }
s3 = {6 7 8 9 10 11 12 13 }
s4 = {1 3 5 7 9 11 13 }
s5 = {2 4 6 8 10 12 13 }

Size of set cover is :3
s3 = {6 7 8 9 10 11 12 13 }
s2 = {2 3 4 5 }
s1 = {1 2 }

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