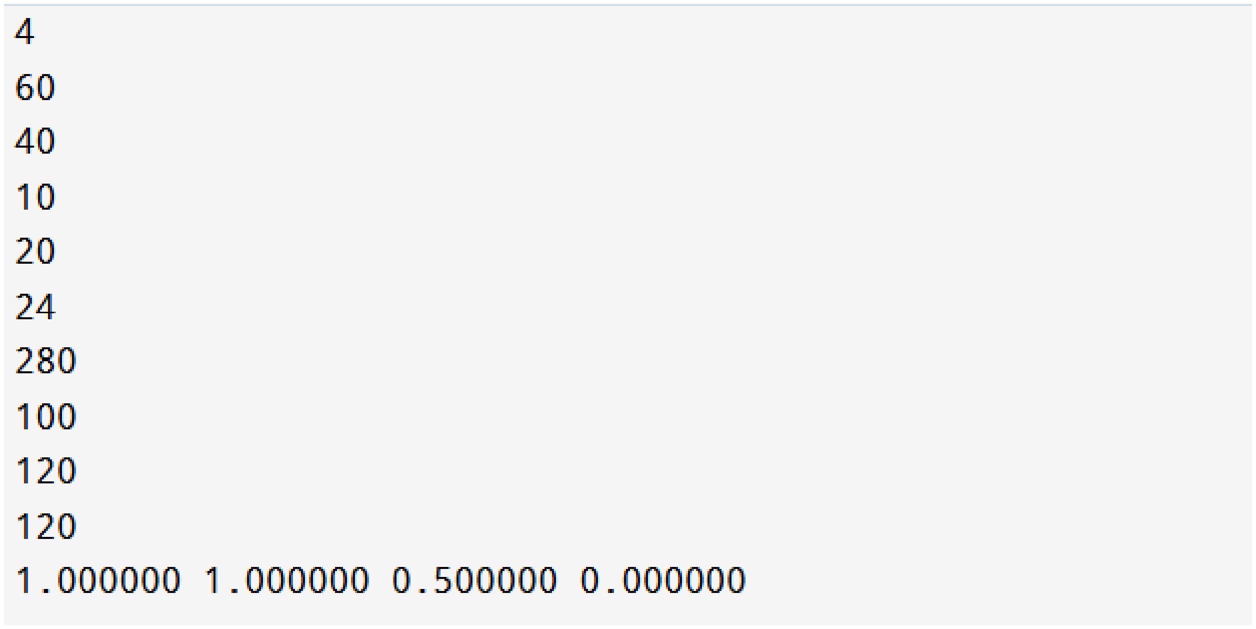
Lab-2: Greedy Algorithms-I

The optimal Knapsack Algorithm:

```
#include <stdio.h>
#include<stdlib.h>
typedef struct knapSack
{
   int weight;
   int value;
   float ratio;
}knapSack;
int compareRatio(const void *a,const void* b)
{
   knapSack *x =(knapSack*)a;
   knapSack *y = (knapSack*)b;
   if(x->ratio<y->ratio) return 1;
   if(y->ratio<x->ratio) return -1;
   return 0;
}
void knps(struct knapSack arr[],int w, int n)
{
   int i =0,weight=0,profit=0;
   float x[n];
   for(int i = 0;i<n;i++)
   {
      x[i] = 0.0;
   }
   while(i<=n && weight < w)
   {
      if(weight + arr[i].weight <= w)
      {
         x[i]=1;
         weight+= arr[i].weight;
         profit+=arr[i].value;
      }
      else
      {
         x[i] = (float)(w-weight)/arr[i].weight;
         weight+= x[i]*arr[i].weight;
```

```
            break;
        }
        i++;
    }
    for(int i =0;i<n;i++)
        printf("%f ",x[i]);
}
int main()
{
    int n;
    scanf("%d",&n);
    int weight;
    scanf("%d",&weight);
    knapSack arr[n];
    int size = sizeof(arr)/sizeof(arr[0]);
    for(int i = 0;i<n;i++)
    {
        scanf("%d",&arr[i].weight);
    }
    for(int i = 0;i<n;i++)
    {
        scanf("%d",&arr[i].value);
    }
    for(int i = 0;i<n;i++)
    {
        arr[i].ratio = (float)arr[i].value/arr[i].weight;
    }
    qsort(arr,n,sizeof(knapSack),compareRatio);
    knps(arr,weight,n);
    return 0;
}
```

O/P:



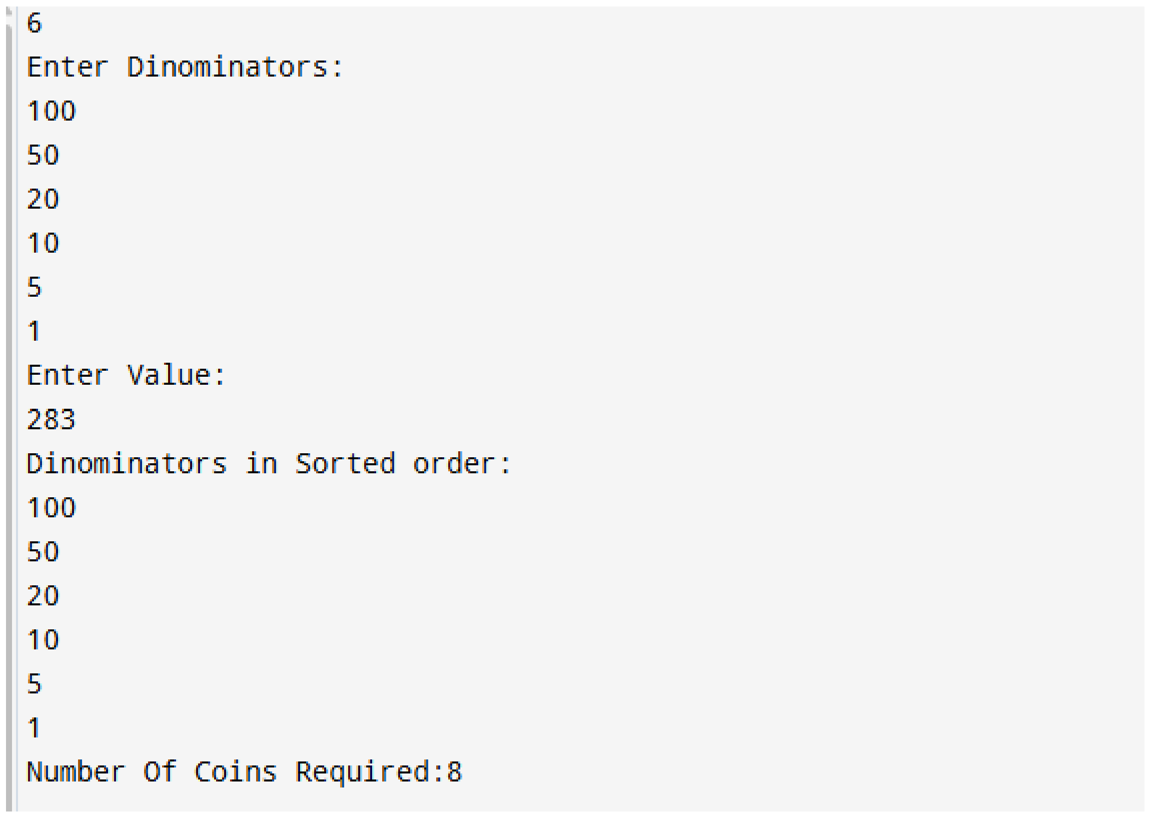
```
4
60
40
10
20
24
280
100
120
120
1.000000 1.000000 0.500000 0.000000
```

MAKE CHANGE:

```
#include <iostream>
using namespace std;
int main() {
   // Write C++ code here
   int n,value;
   cin>>n;
   int d[n];
   cout<<"Enter Dinominators:\n";
   for(int i=0;i<n;i++)
   {
      cin>>d[i];
   }
    cout<<"Enter Value:\n";
   cin>>value;
for(int i=1;i<n;i++)
{
   int j=i;
   while (j > 0 && d[j] > d[j-1]) {
         swap(d[j], d[j-1]);
         j--;
      }
}
cout<<"Dinominators in Sorted order:\n";
 for(int i=0;i<n;i++)
    {
        cout<<d[i]<<" "<<endl;
    }

  int count=0;
  int ans[n]={0};
  for(int i=0;i<n;i++)
  {
      ans[i]=value/d[i];
      value=value%d[i];
      count=count+ans[i];
  }
  cout<<"Number Of Coins Required:"<<count;
    return 0;
}
```

O/P:



```
6
Enter Dinominators:
100
50
20
10
5
1
Enter Value:
283
Dinominators in Sorted order:
100
50
20
10
5
1
Number Of Coins Required:8
```

Activity Selection Problem:

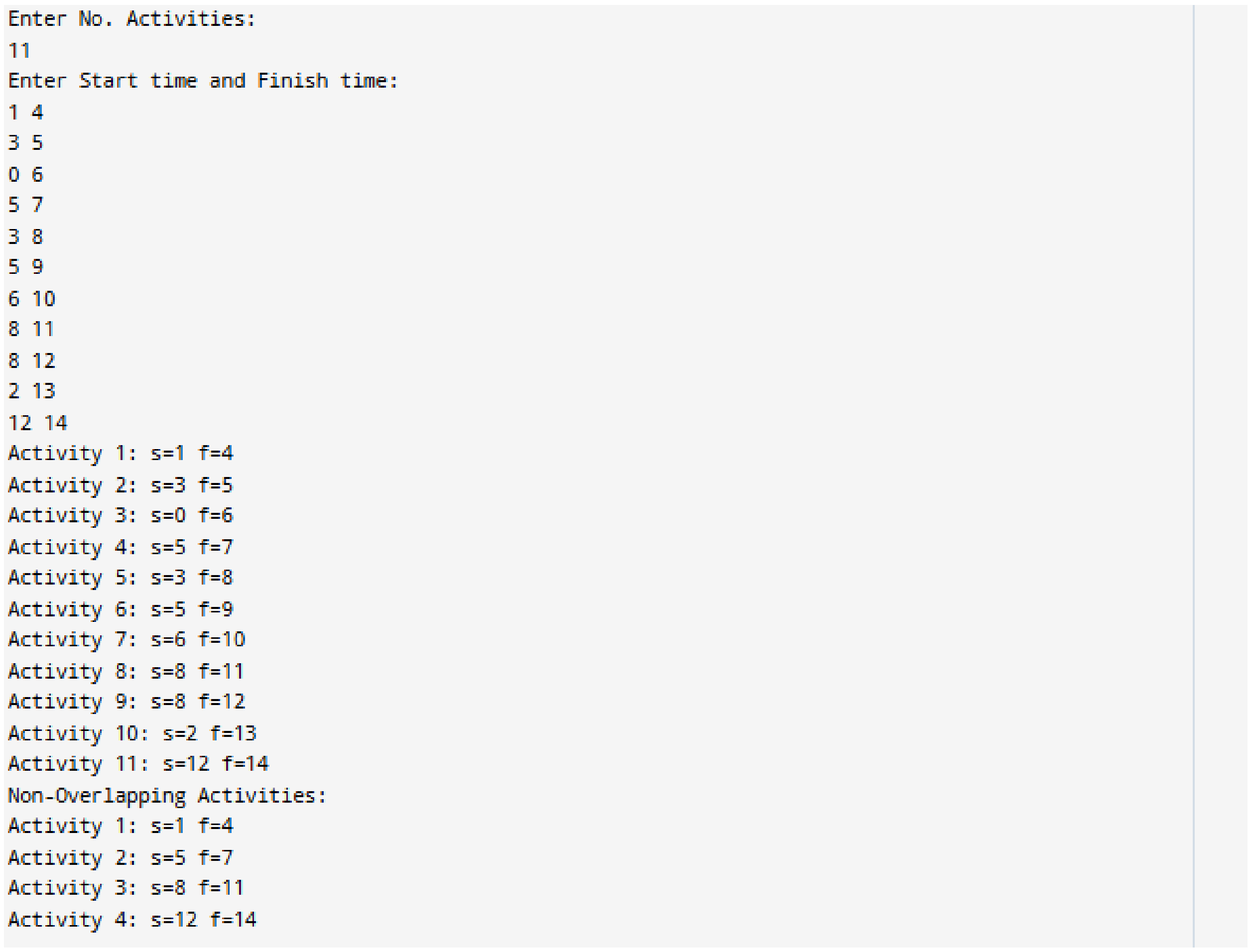
```
#include <iostream>
using namespace std;
struct node {
    int s;
    int f;
};

void ini(node*& item) {
    item = new node;
    cin >> item->s >> item->f;
}
int main() {
   int n;
    cout<<"Enter No. Activities:\n";
    cin>>n;
    node* item[n] = {nullptr};
     cout << "Enter Start time and Finish time:\n";
    for(int i=0;i<n;i++)
    {
        ini(item[i]);
    }
    for (int i = 1; i < n; i++) {
        int j = i;
        while (j > 0 && item[j]->f < item[j-1]->f) {
            swap(item[j], item[j-1]);
            j--;
        }
    }

        for (int i = 0; i < n; i++) {
        cout << "Activity " << i+1 << ": s=" << item[i]->s
             << " f=" << item[i]->f<< endl;
    }
   node* a[n] = {nullptr};
    int i=0;
    a[i]=item[i];
    for(int j=1;j<=n;j++)
    {
        if(item[j]->s>=a[i]->f)
        {
          a[++i]=item[j];
        }
    }
        cout << "Non-Overlapping Activities:\n";


        for (int j = 0; j < i; j++) {
        cout << "Activity " << j+1 << ": s=" << a[j]->s
            << " f=" << a[j]->f<< endl;
    }
    return 0;
}
```

O/P:



```
Enter No. Activities:
11
Enter Start time and Finish time:
1 4
3 5
0 6
5 7
3 8
5 9
6 10
8 11
8 12
2 13
12 14
Activity 1: s=1 f=4
Activity 2: s=3 f=5
Activity 3: s=0 f=6
Activity 4: s=5 f=7
Activity 5: s=3 f=8
Activity 6: s=5 f=9
Activity 7: s=6 f=10
Activity 8: s=8 f=11
Activity 9: s=8 f=12
Activity 10: s=2 f=13
Activity 11: s=12 f=14
Non-Overlapping Activities:
Activity 1: s=1 f=4
Activity 2: s=5 f=7
Activity 3: s=8 f=11
Activity 4: s=12 f=14
```