LAB ASSIGNMENT 2:

Name: Ojaswi Kumar

Section: 2CS11

Write a program to solve the following problems using greedy Greedy approach:

1)Activity selection :

```
#include <iostream>
using namespace std;
void printMaxActivities(int s[], int f[], int n)
{
    int i, j;

    cout << "Following activities are selected" << endl;


    i = 0;
    cout << i << " ";


    for (j = 1; j < n; j++) {

        if (s[j] >= f[i]) {
            cout << j << " ";
            i = j;
        }
    }
```

```
}


int main()
{
    int s[] = { 1, 3, 0, 5, 8, 5 };
    int f[] = { 2, 4, 6, 7, 9, 9 };
    int n = sizeof(s) / sizeof(s[0]);


    printMaxActivities(s, f, n);
    return 0;
}
```

2)Job sequencing:

```
#include <algorithm>
#include <iostream>
using namespace std;

struct Job {

   char id;
   int dead;
   int profit;

};


bool comparison(Job a, Job b)
{
   return (a.profit > b.profit);
}


void printJobScheduling(Job arr[], int n)
{
```

```
   sort(arr, arr + n, comparison);

   int result[n];
   bool slot[n];


   for (int i = 0; i < n; i++)
      slot[i] = false;


   for (int i = 0; i < n; i++) {
      for (int j = min(n, arr[i].dead) - 1; j >= 0; j--) {

         if (slot[j] == false) {
            result[j] = i;
            slot[j] = true;
            break;
         }
      }
   }

   for (int i = 0; i < n; i++)
      if (slot[i])
         cout << arr[result[i]].id << " ";
}

int main()
{
   Job arr[] = { { 'a', 2, 100 },
            { 'b', 1, 19 },
            { 'c', 2, 27 },
            { 'd', 1, 25 },
            { 'e', 3, 15 } };

   int n = sizeof(arr) / sizeof(arr[0]);
   cout << "Following is maximum profit sequence of jobs "
         "\n";


   printJobScheduling(arr, n);
```

3)Fractional knapsack:

```cpp
// C++ program to solve fractional Knapsack Problem

#include <bits/stdc++.h>
using namespace std;


struct Item {
    int value, weight;


    Item(int value, int weight)
    {
        this->value = value;
        this->weight = weight;
    }
};


static bool cmp(struct Item a, struct Item b)
{
    double r1 = (double)a.value / (double)a.weight;
    double r2 = (double)b.value / (double)b.weight;
    return r1 > r2;
}
```

```
double fractionalKnapsack(int W, struct Item arr[], int
N)
{

    sort(arr, arr + N, cmp);

    double finalvalue = 0.0;


    for (int i = 0; i < N; i++) {



        if (arr[i].weight <= W) {
            W -= arr[i].weight;
            finalvalue += arr[i].value;
        }

        else {
            finalvalue
                += arr[i].value
                * ((double)W / (double)arr[i].weight);
            break;
        }
    }


    return finalvalue;
}


int main()
```

```
{
    int W = 50;
    Item arr[] = { { 60, 10 }, { 100, 20 }, { 120, 30 }
};

    int N = sizeof(arr) / sizeof(arr[0]);


    cout << fractionalKnapsack(W, arr, N);
    return 0;
}
```

ADDITIONAL QUESTIONS :

1)

Given an array of size N, find the majority element. The majority element is the element that appears more than floor(N/2) times.
You may assume that the array is non-empty and the majority element always exist in the array.

```
#include <iostream>

#include <cmath>
```

```
int majorityElement(int arr[], int size) {
    int count = 1;
    int majority = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] == majority)
            count++;
        else {
            count--;
            if (count == 0) {
                majority = arr[i];
                count = 1;
            }
        }
    }
    count = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i] == majority)
            count++;
    }
    if (count > floor(size / 2))
```

```
        return majority;
    else
        return -1;
}

int main() {
    int arr[] = {2, 2, 3, 5, 2, 2, 6};
    int size = sizeof(arr) / sizeof(arr[0]);
    int result = majorityElement(arr, size);
    if (result != -1)
        std::cout << "The majority element is: " <<
result << std::endl;
    else
        std::cout << "There is no majority element." <<
std::endl;
    return 0;
}
```