Student Name:- CHAUDHARY HAMDAN

Student Roll No.:- 1905387

Algorithm Lab. Class Assignment-10

CSE Group 1

Date: - 1st October 2021

1. **Write a program to implementation of Fractional Knapsack algorithm.**

Program

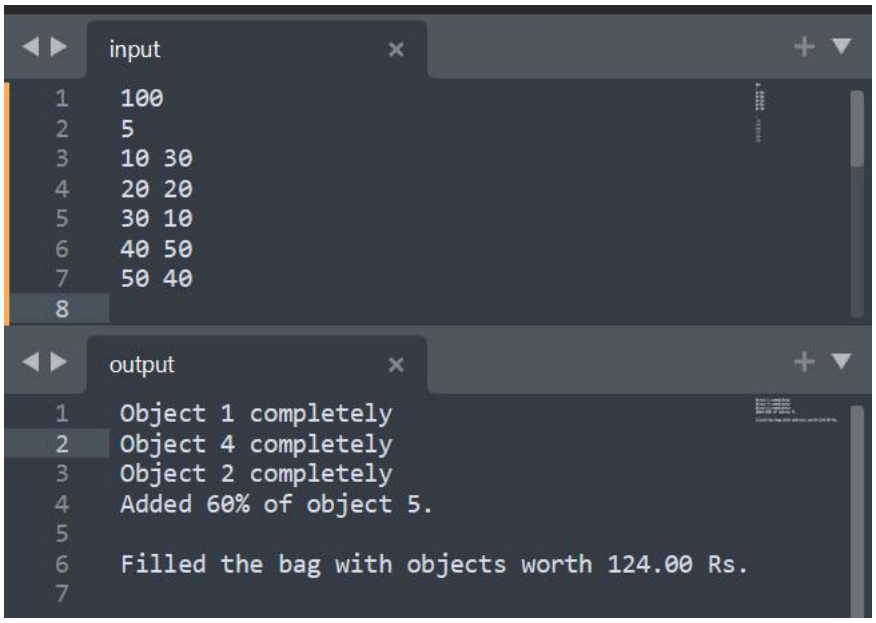
```
// Author: Chaudhary Hamdan
// Generated: Fri Oct 1 12:22:06 2021

#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define sf(x)          scanf("%d", &x)
#define pf             printf
#define pfs(x)         printf("%d ", x)
#define pfn(x)         printf("%d\n", x)
#define pfc(x)         printf("%d, ", x)
#define FI(i,x,y,inc)   for(int i = x; i < y; i += inc)
#define F(i,x,y)       FI(i, x, y, 1)
#define F0(i,n)        FI(i, 0, n, 1)
#define RF(i,x,y)       for(int i = x; i >= y; i--)
#define pfarr(i,a,n)    for(int i = 0; i < n-1; i++) pfs(a[i]); pfn(a[n-1]);



void i_o_from_file();
```

```
int main() {

    i_o_from_file();

    /* ********************************************** */

    int capacity, no_items, cur_weight, item;
    int used[10];
    float total_profit;
    int i;
    int weight[10];
    int value[10];

    sf(capacity);

    sf(no_items);

    F0(i, no_items) {
        sf(weight[i]);
        sf(value[i]);
    }

    F0(i, no_items) {
        used[i] = 0;
    }

    cur_weight = capacity;
```

```
    while (cur_weight > 0)
    {
        item = -1;
        F0(i, no_items) {
            if ((used[i] == 0) && ((item == -1) || (value[i] * 1.0 / weight[i] > value[item]
* 1.0 / weight[item])))
                item = i;
        }

        used[item] = 1;
        cur_weight -= weight[item];
        total_profit += value[item];

        if (cur_weight >= 0) {
            printf("Object %d completely\n", item + 1);
        }
        else {

            int item_percent = (int) ((1 + cur_weight * 1.0 / weight[item]) * 100);

            pf("Added %d%% of object %d.\n", item_percent, item + 1);

            total_profit -= value[item];
            total_profit += (1 + cur_weight * 1.0 / weight[item]) * value[item];
        }
    }

    pf("\nFilled the bag with objects worth %.2f Rs.\n", total_profit);

    return 0;
}
```

```
void i_o_from_file() {

#ifndef ONLINE_JUDGE
    freopen("C:\\Users\\KIIT\\input", "r", stdin);
    freopen("C:\\Users\\KIIT\\output", "w", stdout);
#endif
}
```

Output



2. **Write a program to implement the activity-selection problem stated as follows: You are given n activities with their start and finish times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time. Example: Consider the following 6 activities. start[] = {1, 3, 0, 5, 8, 5}; finish[] = {2, 4, 6, 7, 9, 9}; The maximum set of activities that can be executed by a single person is {0, 1, 3, 4}.**

Program

```
// Author: Chaudhary Hamdan

#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define sf(x)          scanf("%d", &x)
#define pf             printf
#define pfs(x)         printf("%d ", x)
#define pfn(x)         printf("%d\n", x)
#define pfc(x)         printf("%d, ", x)
#define F(i,x,y)       FI(i, x, y, 1)
#define F0(i,n)        FI(i, 0, n, 1)

void i_o_from_file();

void activitySelection(int s[], int f[], int n)
{
    int i, j;

    pf("Activities selected:-\n");
```

```
    i = 0;
    pfs(i + 1);

    for (j = 1; j < n; j++) {
        if (s[j] >= f[i]) {
            pfs(j + 1);
            i = j;
        }
    }
}
int main() {
    i_o_from_file();
    /* ********************************************** */
    int n;
    sf(n);

    int s[n], f[n];

    F0(i, n) {
        sf(s[i]);
        sf(f[i]); // Giving input in sorted form wrt finish times.
    }

    time_t start, end;
    double time;
    start = clock();

    activitySelection(s, f, n);

    end = clock();
```

```
    time = (end - start) * 1.0 / CLOCKS_PER_SEC;

    pf("\nTime: %f\n", time);
    return 0;
}

void i_o_from_file() {

#ifndef ONLINE_JUDGE
    freopen("C:\\Users\\KIIT\\input", "r", stdin);
    freopen("C:\\Users\\KIIT\\output", "w", stdout);
#endif
}
```

Output