**SE COMPS A BATCH-C**

**9913 Mark Lopes**

**Fractional Knapsack:-**

#include <stdlib.h>

#include <stdio.h>

struct Item

{

    int item\_no;

    int profit;

    float weight;

    float ratio;

} arr[50], arr1[50];

void sort(struct Item \*arr1, int n)

{

    int swapped = 1;

    for (int pass = 1; pass <= n - 1 && swapped == 1; pass++)

    {

        swapped = 0;

        for (int j = 1; j <= n - pass; j++)

        {

            if (arr1[j].ratio < arr1[j + 1].ratio)

            {

                struct Item temp = arr1[j];

                arr1[j] = arr1[j + 1];

                arr1[j + 1] = temp;

                swapped = 1;

            }

        }

    }

}

void fractionalKnapsack(struct Item \*arr, int n, int M)

{

    float total = 0;

    float x[50];

    int u = M;

    int i = 1;

    printf("\nAfter sorting the elements in Profit/weight manner:-");

    while (u != 0 && i <= n)

    {

        if (arr[i].weight <= u)

        {

            u = u - arr[i].weight;

            total = total + arr[i].profit;

            x[arr[i].item\_no] = 1;

        }

        else

        {

            x[arr[i].item\_no] = u / arr[i].weight;

            total = total + (arr[i].profit \* x[arr[i].item\_no]);

            u = 0;

        }

        printf("\nRemaining capacity after sorted element %d: %d\n", i, u);

        i++;

    }

    printf("Total profit is %0.2f\n", total);

    printf("The solution vector is: ");

    for (i = 1; i <= n; i++)

    {

        printf("%0.2f ", x[i]);

    }

}

int main()

{

    int M, n, i;

    printf("Enter the total capacity: ");

    scanf("%d", &M);

    printf("Enter the number of items: ");

    scanf("%d", &n);

    for (i = 1; i <= n; i++)

    {

        arr[i].item\_no = i;

        printf("Enter the profit and weight for item %d: ", i);

        scanf("%d %f", &arr[i].profit, &arr[i].weight);

    }

    for (i = 1; i <= n; i++)

    {

        arr[i].ratio = arr[i].profit / arr[i].weight;

    }

    for (i = 1; i <= n; i++)

    {

        arr1[i] = arr[i];

    }

    sort(arr1, n);

    printf("The sorted ratios: \n");

    for (i = 1; i <= n; i++)

    {

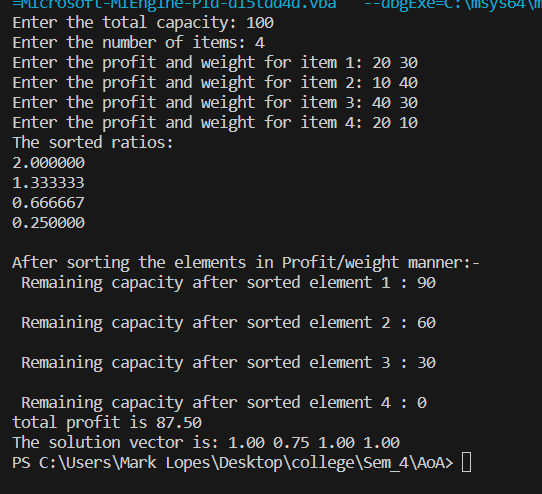
        printf("%f\n", arr1[i].ratio);

    }

    fractionalKnapsack(arr1, n, M);

    return 0;

}

****

**Activity selection problem:-**

import java.util.\*;

class Solution {

    public static class Pair implements Comparable<Pair> {

        int start;

        int end;

        public Pair(int s, int e) {

            this.start = s;

            this.end = e;

        }

        public int compareTo(Pair o) {

            if (this.end < o.end) {

                return -1;

            } else if (this.end > o.end) {

                return 1;

            } else {

                return 0;

            }

        }

    }

    public static int activitySelection(int start[], int end[], int n) {

        Pair[] activities = new Pair[n];

        for (int i = 0; i < n; i++) {

            activities[i] = new Pair(start[i], end[i]);

        }

        Arrays.sort(activities);

        int ans = 0;

        int lastActivityEnd = 0;

        for (int i = 0; i < n; i++) {

            Pair activity = activities[i];

            if (activity.start > lastActivityEnd) {

                lastActivityEnd = activity.end;

                ans++;

            }

        }

        return ans;

    }

    public static void main(String[] args) {

        int[] startTimes = {2, 1};

        int[] endTimes = {2, 2};

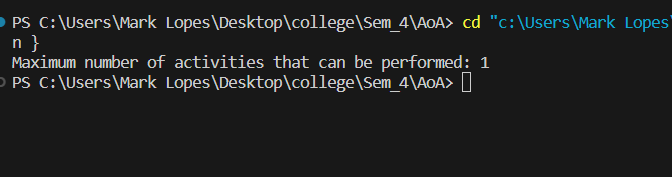
        int n = startTimes.length;

        int maxActivities = activitySelection(startTimes, endTimes, n);

        System.out.println("Maximum number of activities that can be performed: " + maxActivities);

    }

}

****