ASSIGNMENT 1

NAME: Mujeeb ur Rahman khan

Roll no: 20736

CODE

/\*

Write a program to implement Fractional knapsack using Greedy algorithm and 0/1 knapsack using dynamic programming.

Show that Greedy strategy does not necessarily yield an optimal solution over a dynamic programming approach.

\*/

#include <bits/stdc++.h>

using namespace std;

int mainMenu();

int fractionalKnapscak(vector<int> wt, vector<int> val, int W, int n);

int zeroOneKnapscak(vector<int> wt, vector<int> val, int W, int n);

int main() {

    int n,len,W,ans;

    while(1) {

        cout<<endl<<"Enter length of array ";

            cin>>len;

        vector<int> values(len);

        vector<int> weight(len);

        cout<<endl<<"Enter values ";

            for(int i=0;i<len;i++)

                cin>>values[i];

        cout<<endl<<"Enter weight ";

            for(int i=0;i<len;i++)

                cin>>weight[i];

        cout<<endl<<"Enter Capacity of bag ";

            cin>>W;

        n = mainMenu();

        switch (n)

        {

        case 1:

            ans = fractionalKnapscak(weight,values,W,len);

            cout<<endl<<"maximum value is "<<ans<<endl;

            break;

        case 2:

            ans = zeroOneKnapscak(weight,values,W,len-1);

            cout<<endl<<"maximum value is "<<ans<<endl;

            break;

        case 3:

            exit(0);

            break;

        default:

            cout<<endl<<"enter valid choice ";

            break;

        }

    }

    return 0;

}

int mainMenu() {

    cout<<endl<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

    cout<<endl<<"\t\t 1. fractional Knapsack";

    cout<<endl<<"\t\t 2. 0/1 knapsack";

    cout<<endl<<"\t\t 3. exit";

    cout<<endl<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

    int n;

    cout<<endl<<"Enter your choice ";

        cin>>n;

    return n;

}

int fractionalKnapscak(vector<int> wt, vector<int> val, int W, int n) {

    vector<int> ratio(n);

    for(int i=0;i<n;i++)

        ratio[i] = val[i]/wt[i];

    sort(ratio.begin(), ratio.end() , greater<int>());

    int value = 0;

    int capacity = 0;

    for(int i=0;i<n;i++) {

        if(capacity+wt[i]<=W) {

            capacity+=wt[i];

            value+=val[i];

        }else{

            int temp = W-capacity;

            int x = temp\*ratio[i];

            value+=x;

            break;

        }

    }

    return value;

}

int zeroOneKnapscak(vector<int> wt, vector<int> val, int W, int n) {

    if(n==0) {

        if(W-wt[n]>=0){

            return val[0];

        }else{

            return 0;

        }

    }

    int include = 0;

        if(W-wt[n]>=0){

            include = val[n] + zeroOneKnapscak(wt,val,W-wt[n],n-1);

        }

        int exclude = 0 + zeroOneKnapscak(wt,val,W,n-1);

        return max(include,exclude);

}

OUTPUT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter length of array 3

Enter values 60 100 120

Enter weight 10 20 30

Enter Capacity of bag 50

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                 1. fractional Knapsack

                 2. 0/1 knapsack

                 3. exit

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter your choice 1

maximum value is 240

Enter length of array 3

Enter values 60 100 120

Enter weight 10 20 30

Enter Capacity of bag 50

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                 1. fractional Knapsack

                 2. 0/1 knapsack

                 3. exit

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter your choice 2

maximum value is 220

Enter length of array