**NAME: PARTH NITESHKUMAR PATEL**

**ID: 19DCS098**

**CONTACT: 8980146044**

**BATCH: A**

**SUBJECT: DESIGN AND ANALYSIS OF ALGORITHM**

**SUBJECT CODE: CS 351**

**PRACTICAL-1**

**AIM:**

Implement Knapsack Problem using Greedy Approach

**PROGRAM CODE:**

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

typedef struct {

   double v;

   double w;

} Item;

void input(Item items[],int sizeOfItems) {

   cout<<"-----------------------------------------------------------------------------------"<<endl;

   cout << "ENTER THE TOTAL "<< sizeOfItems <<" ITEM'S VALUES/PROFITS AND WEIGHTS : " <<endl;

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

      cout<<"-----------------------------------------------------------------------------------"<<endl;

      cout << "ENTER V : "<<i+1<<" : ";

      cin >> items[i].v;

      cout << "ENTER W : "<< i+1 << " : ";

      cin >> items[i].w;

   }

}

bool compare(Item a, Item b) {

    double r1 = (double)(a.v / a.w);

    double r2 = (double)(b.v / b.w);

    return r1 > r2;

}

void display(Item items[], int sizeOfItems) {

   int i;

   cout<<"-----------------------------------------------------------------------------------"<<endl;

   cout << "values: ";

   for(i = 0; i < sizeOfItems; i++) {

      cout << items[i].v << "\t";

   }

   cout<<endl;

   cout<<"-----------------------------------------------------------------------------------"<<endl;

   cout << endl << "weight: ";

   for (i = 0; i < sizeOfItems; i++) {

      cout << items[i].w << "\t";

   }

   cout << endl;

}

double knapsack(Item items[], int sizeOfItems, int W) {

   int i, j;

   double totalValue = 0, totalWeight = 0;

   cout<<"-----------------------------------------------------------------------------------"<<endl;

   cout<<"PROFIT PER UNIT WEIGHT :\n";

   cout<<"-----------------------------------------------------------------------------------"<<endl;

   cout<<"Value     Weight    Profit\n";

   cout<<"-----------------------------------------------------------------------------------"<<endl;

   for (int i = 0; i < sizeOfItems; i++)

   {

      cout << items[i].v << "         " << items[i].w << "         "

         << ((double)items[i].v / items[i].w) << endl;

   }

   sort(items, items+sizeOfItems, compare);

   for(i=0; i<sizeOfItems; i++) {

      if(totalWeight + items[i].w<= W) {

         totalValue += items[i].v ;

         totalWeight += items[i].w;

      } else {

         int wt = W-totalWeight;

         totalValue += items[i].v\*((double)wt / items[i].w);

         totalWeight += wt;

         break;

      }

   }

   cout<<"-----------------------------------------------------------------------------------"<<endl;

   cout << "TOTAL WEIGHT IN THE BAG: " << totalWeight<<endl;

   return totalValue;

}

int main() {

   int W,n;

   cout<<"-----------------------------------------------------------------------------------"<<endl;

   cout<<"ENTER THE TOTAL NUMBER OF ITEMS:";

   cin>>n;

   Item items[n];

   input(items, n);

   cout<<"-----------------------------------------------------------------------------------"<<endl;

   cout << "DATA :\n";

   display(items,n);

   cout<<"-----------------------------------------------------------------------------------"<<endl;

   cout<< "ENTER THE KNAPSACK WEIGHT: \n";

   cin >> W;

   double mxVal = knapsack(items, n, W);

   cout<<"-----------------------------------------------------------------------------------"<<endl;

   cout << "MAXIMUM PROFIT FOR "<< W <<" WEIGHT : "<< mxVal;

   cout<<endl;

   cout<<"-----------------------------------------------------------------------------------"<<endl;

   cout<<"PARTH PATEL\n19DCS098"<<endl;

   cout<<"-----------------------------------------------------------------------------------"<<endl;

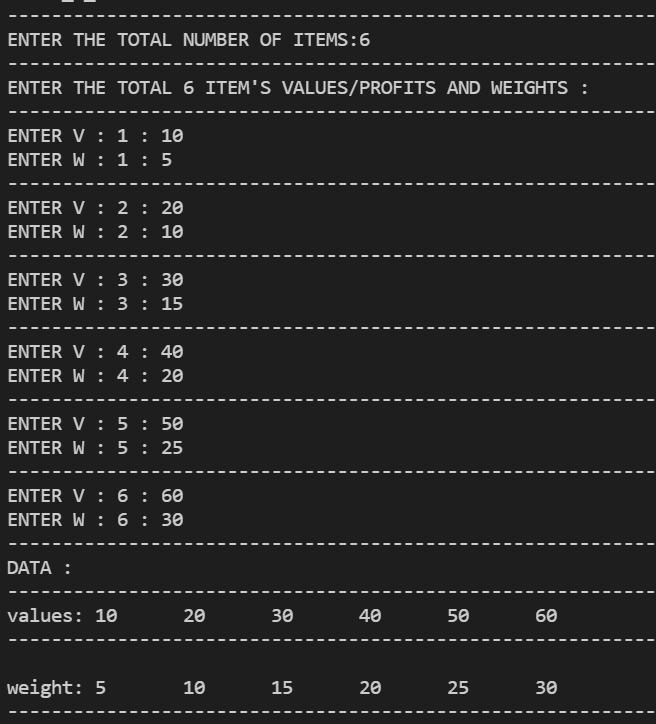
   cout<<"CS 351 DAA EXTERNAL PRACTICAL EXAM"<<endl;

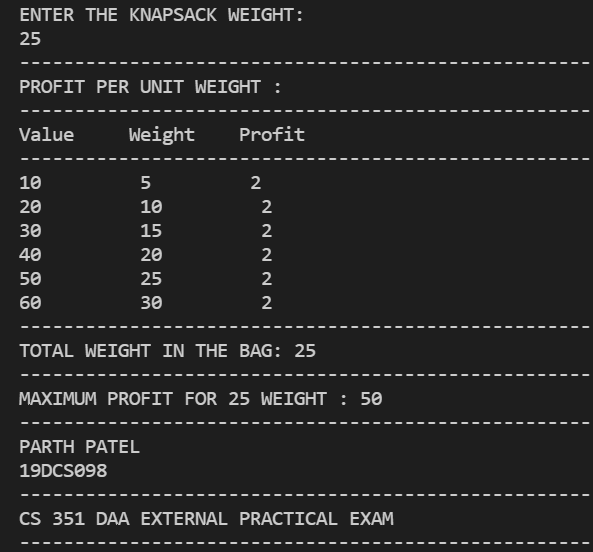
   cout<<"-----------------------------------------------------------------------------------"<<endl;

   return 0;

}

**OUTPUT:**





**PRACTICAL-2**

**AIM:**

Implement Matrix Chain Multiplication using Dynamic Programming

**PROGRAM CODE:**

#include <bits/stdc++.h>

using namespace std;

int MatrixChainMultiplication(int product[], int n)

{

    int matrix[n][n];

    int i, j, k, L, q;

    for (i = 1; i < n; i++)

        matrix[i][i] = 0;

    for (L = 2; L < n; L++)

    {

        for (i = 1; i < n - L + 1; i++)

        {

            j = i + L - 1;

            matrix[i][j] = INT\_MAX;

            for (k = i; k <= j - 1; k++)

            {

                q = matrix[i][k] + matrix[k + 1][j] +

                    product[i - 1] \* product[k] \* product[j];

                if (q < matrix[i][j])

                    matrix[i][j] = q;

            }

        }

    }

    return matrix[1][n - 1];

}

int main()

{

    int n;

    cout<<"-------------------------------------------"<<endl;

    cout << "ENTER THE TOTAL NUMBER OF MATRICES : ";

    cin >> n;

    int arr[n];

    cout<<"-------------------------------------------"<<endl;

    for (int i = 0; i < n; i++)

    {

        cout << "ENTER THE NxN DIMENSIONS OF MATRIX -> " << i << " : ";

        cin >> arr[i];

        cout<<"-------------------------------------------"<<endl;

    }

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

    cout<<"-------------------------------------------"<<endl;

    cout << "MINIMUM NUMBER OF MULTIPLICATIONS NEEDED : " << MatrixChainMultiplication(arr, length) << endl;

    cout<<"-------------------------------------------"<<endl;

    cout << "PARTH PATEL\n19DCS098" << endl;

    cout<<"-------------------------------------------"<<endl;

    cout<<"[CS 351] DAA EXTERNAL PRACTICAL EXAM"<<endl;

    cout<<"-------------------------------------------"<<endl;

    return 0;

}

**OUTPUT:**

