**Motilal Nehru National Institute of Technology, Allahabad**

**(Department of Computer Science & Engineering)**

**(MCA- 3rd Sem)**

**Assignment-10**

**Design Analysis & Algorithms**

**NAME : SHISHU REG:2020CA89**

1. Write a c program to implement Fractional knapsack problem

#include <stdio.h>

void knapsack(int n, float weight[], float profit[], float capacity)

{

    float x[20], tp = 0;

    int i, j, u;

    u = capacity;

    for (i = 0; i < n; i++)

        x[i] = 0.0;

    for (i = 0; i < n; i++)

    {

        if (weight[i] > u)

            break;

        else

        {

            x[i] = 1.0;

            tp = tp + profit[i];

            u = u - weight[i];

        }

    }

    if (i < n)

        x[i] = u / weight[i];

    tp = tp + (x[i] \* profit[i]);

    printf("\nThe result vector is:- ");

    for (i = 0; i < n; i++)

        printf("%f\t", x[i]);

    printf("\nMaximum profit is:- %f", tp);

}

int main()

{

    float weight[20], profit[20], capacity;

    int num, i, j;

    float ratio[20], temp;

    printf("\nEnter the no. of objects:- ");

    scanf("%d", &num);

    printf("\nEnter the wts and profits of each object:- ");

    for (i = 0; i < num; i++)

    {

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

    }

    printf("\nEnter the capacityacity of knapsack:- ");

    scanf("%f", &capacity);

    for (i = 0; i < num; i++)

    {

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

    }

    for (i = 0; i < num; i++)

    {

        for (j = i + 1; j < num; j++)

        {

            if (ratio[i] < ratio[j])

            {

                temp = ratio[j];

                ratio[j] = ratio[i];

                ratio[i] = temp;

                temp = weight[j];

                weight[j] = weight[i];

                weight[i] = temp;

                temp = profit[j];

                profit[j] = profit[i];

                profit[i] = temp;

            }

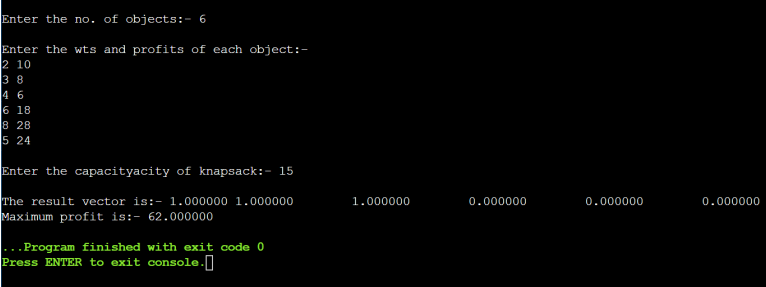
        }

    }

    knapsack(num, weight, profit, capacity);

    return (0);

}



2.Write a C program to implement [Huffman Coding](https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/)

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

typedef struct node

{

    char ch;

    int freq;

    struct node \*left;

    struct node \*right;

} node;

node \*heap[100];

int heapSize = 0;

void Insert(node \*element)

{

    heapSize++;

    heap[heapSize] = element;

    int now = heapSize;

    while (heap[now / 2]->freq > element->freq)

    {

        heap[now] = heap[now / 2];

        now /= 2;

    }

    heap[now] = element;

}

node \*DeleteMin()

{

    node \*minElement, \*lastElement;

    int child, now;

    minElement = heap[1];

    lastElement = heap[heapSize--];

    for (now = 1; now \* 2 <= heapSize; now = child)

    {

        child = now \* 2;

        if (child != heapSize && heap[child + 1]->freq < heap[child]->freq)

        {

            child++;

        }

        if (lastElement->freq > heap[child]->freq)

        {

            heap[now] = heap[child];

        }

        else

        {

            break;

        }

    }

    heap[now] = lastElement;

    return minElement;

}

void print(node \*temp, char \*code)

{

    if (temp->left == NULL && temp->right == NULL)

    {

        printf("char %c code %s\n", temp->ch, code);

        return;

    }

    int length = strlen(code);

    char leftcode[10], rightcode[10];

    strcpy(leftcode, code);

    strcpy(rightcode, code);

    leftcode[length] = '0';

    leftcode[length + 1] = '\0';

    rightcode[length] = '1';

    rightcode[length + 1] = '\0';

    print(temp->left, leftcode);

    print(temp->right, rightcode);

}

int main()

{

    heap[0] = (node \*)malloc(sizeof(node));

    heap[0]->freq = 0;

    int n;

    printf("Enter the no of characters: ");

    scanf("%d", &n);

    printf("Enter the characters and their frequencies: ");

    char ch;

    int freq, i;

    for (i = 0; i < n; i++)

    {

        scanf(" %c", &ch);

        scanf("%d", &freq);

        node \*temp = (node \*)malloc(sizeof(node));

        temp->ch = ch;

        temp->freq = freq;

        temp->left = temp->right = NULL;

        Insert(temp);

    }

    if (n == 1)

    {

        printf("char %c code 0\n", ch);

        return 0;

    }

    for (i = 0; i < n - 1; i++)

    {

        node \*left = DeleteMin();

        node \*right = DeleteMin();

        node \*temp = (node \*)malloc(sizeof(node));

        temp->ch = 0;

        temp->left = left;

        temp->right = right;

        temp->freq = left->freq + right->freq;

        Insert(temp);

    }

    node \*tree = DeleteMin();

    char code[10];

    code[0] = '\0';

    print(tree, code);

}

