Name: Atharva Bilonikar

Uid No: 2021700011

CSE DS D1

DAA exp 8

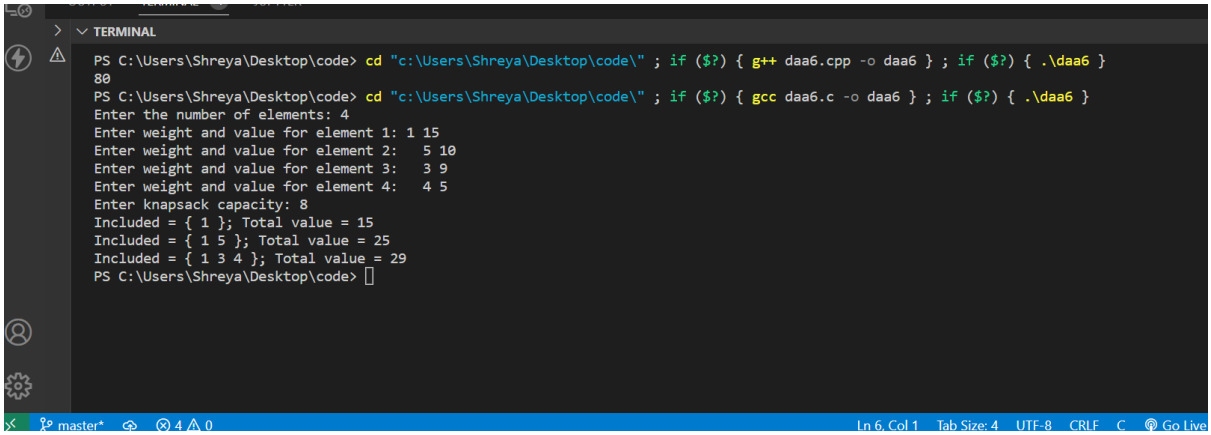
Aim: Branch and bound (To implement 0/1 Knapsack problem using Branch and Bound.)

Code :

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef enum { NO, YES } BOOL;

int N;
int vals[100];
int wts[100];

int cap = 0;
int mval = 0;

void getWeightAndValue (BOOL incl[N], int *weight, int *value) {
    int i, w = 0, v = 0;
    for (i = 0; i < N; ++i) {
        if (incl[i]) {
            w += wts[i];
            v += vals[i];
        }
    }
    *weight = w;
    *value = v;
}

void printSubset (BOOL incl[N]) {
    int i;
    int val = 0;
    printf("Included = { ");
    for (i = 0; i < N; ++i) {
        if (incl[i]) {
            printf("%d ", wts[i]);
            val += vals[i];
        }
```

```
    }
    printf("}; Total value = %d\n", val);
}

void findKnapsack (BOOL incl[N], int i) {
    int cwt, cval;
    getWeightAndValue(incl, &cwt, &cval);
    if (cwt <= cap) {
        if (cval > mval) {
            printSubset(incl);
            mval = cval;
        }
    }
    if (i == N || cwt >= cap) {
        return;
    }
    int x = wts[i];
    BOOL use[N], nouse[N];
    memcpy(use, incl, sizeof(use));
    memcpy(nouse, incl, sizeof(nouse));
    use[i] = YES;
    nouse[i] = NO;
    findKnapsack(use, i+1);
    findKnapsack(nouse, i+1);
}

int main(int argc, char const * argv[]) {
    printf("Enter the number of elements: ");
    scanf(" %d", &N);
    BOOL incl[N];
    int i;
    for (i = 0; i < N; ++i) {
        printf("Enter weight and value for element %d: ", i+1);
        scanf(" %d %d", &wts[i], &vals[i]);
        incl[i] = NO;
    }
    printf("Enter knapsack capacity: ");
    scanf(" %d", &cap);
    findKnapsack(incl, 0);
    return 0;
}
```

Output :



Conclusion :

In this experiment , I was able to implement 0/1 knapsack problem using branch and bound