RBU, Nagpur
CSE III Sem
PRACTICAL NO. 4

Name:	**Kishan Saini**
Sec-Batch-Rollno:	**A4-B4-55**

Knapsack Implementation

Task A:

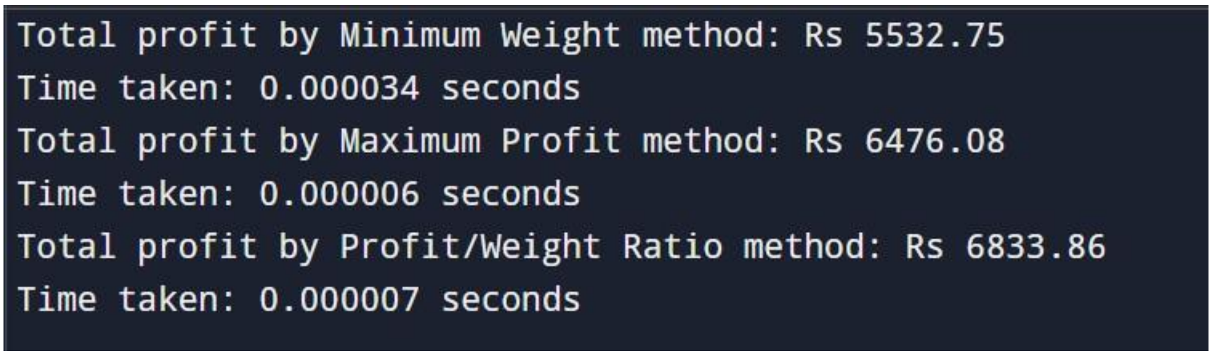
```
#define N 50
#define CAPACITY 850
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int weights[N] = {7, 0, 30, 22, 80, 94, 11, 81, 70, 64, 59, 18, 0,
36, 3, 8, 15, 42, 9, 0, 42, 47, 52, 32, 26, 48, 55, 6, 29, 84, 2, 4,
18, 56, 7, 29, 93, 44, 71, 3, 86, 66, 31, 65, 0, 79, 20, 65, 52,
13};
int profits[N] = {360, 83, 59, 130, 431, 67, 230, 52, 93, 125, 670,
892, 600, 38, 48, 147, 78, 256, 63, 17, 120, 164, 432, 35, 92, 110,
22, 42, 50, 323, 514, 28, 87, 73, 78, 15, 26, 78, 210, 36, 85, 189,
274, 43, 33, 10, 19, 389, 276, 312};

typedef struct {
    int index;
    int weight;
    int profit;
    double ratio;
} Box;

int cmp_min_weight(const void *a, const void *b) {
    Box *boxA = (Box *)a;
    Box *boxB = (Box *)b;
    return boxA->weight - boxB->weight;
}

int cmp_max_profit(const void *a, const void *b) {
    Box *boxA = (Box *)a;
    Box *boxB = (Box *)b;
    return boxB->profit - boxA->profit;
}
```

```
int cmp_max_ratio(const void *a, const void *b) { Box *boxA = (Box
        *)a;
        Box *boxB = (Box *)b;
        if (boxB->ratio > boxA->ratio) return 1;
        else if (boxB->ratio < boxA->ratio) return -1; else return 0;
}

double knapsack(Box boxes[], int n, int capacity) { double x[n];
        for (int i = 0; i < n; i++) { x[i] = 0.0;
        }
        int current_weight = 0; int i = 0;
        while (current_weight < capacity && i < n) { if
                (boxes[i].weight == 0) {
                        i++;
                        continue;
                }
                if (current_weight + boxes[i].weight <= capacity) { x[i] = 1.0;
                        current_weight += boxes[i].weight;
                } else {
                        x[i] = (double)(capacity - current_weight) / boxes[i].weight;
                        current_weight = capacity;
                } i++;
        }
        double total_profit = 0.0; for (int j = 0; j
        < n; j++) {
                total_profit += boxes[j].profit * x[j];
        }
        return total_profit;
}

void copy_boxes(Box dest[], Box src[], int n) { for (int i = 0; i < n;
        i++) {
                dest[i] = src[i];
        }
}
```

```
int main() {
        Box boxes[N];
        for (int i = 0; i < N; i++) { boxes[i].index = i;
                boxes[i].weight = weights[i]; boxes[i].profit =
                profits[i];
                boxes[i].ratio = (weights[i] == 0) ? 0.0 : (double)profits[i] /
weights[i];
        }

        Box temp[N]; double
        profit;
        clock_t start, end;

        copy_boxes(temp, boxes, N); start =
        clock();
        qsort(temp, N, sizeof(Box), cmp_min_weight); profit =
        knapsack(temp, N, CAPACITY);
        end = clock();
        printf("Total profit by Minimum Weight method: Rs %.2f\n", profit);
        printf("Time taken: %f seconds\n", (double)(end - start) / CLOCKS_PER_SEC);

        copy_boxes(temp, boxes, N); start =
        clock();
        qsort(temp, N, sizeof(Box), cmp_max_profit); profit =
        knapsack(temp, N, CAPACITY);
        end = clock();
        printf("Total profit by Maximum Profit method: Rs %.2f\n", profit);
        printf("Time taken: %f seconds\n", (double)(end - start) / CLOCKS_PER_SEC);

        copy_boxes(temp, boxes, N); start =
        clock();
        qsort(temp, N, sizeof(Box), cmp_max_ratio); profit =
        knapsack(temp, N, CAPACITY);
        end = clock();
        printf("Total profit by Profit/Weight Ratio method: Rs %.2f\n", profit);
```

```c
    printf("Time taken: %f seconds\n", (double)(end - start) /
CLOCKS_PER_SEC);

    return 0;
}
```

OUTPUT:



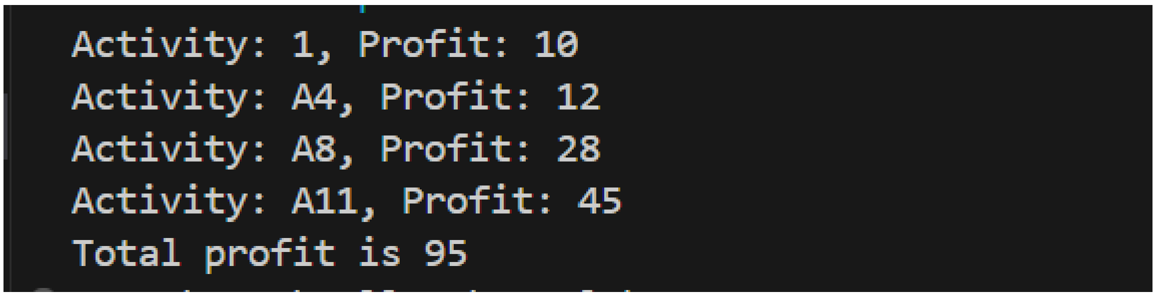
```
Total profit by Minimum Weight method: Rs 5532.75
Time taken: 0.000034 seconds
Total profit by Maximum Profit method: Rs 6476.08
Time taken: 0.000006 seconds
Total profit by Profit/Weight Ratio method: Rs 6833.86
Time taken: 0.000007 seconds
```

Task B:

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

struct Job {
    int start;
    int finish;
    int profit;
};

int ASP(struct Job jobs[]) {
    int totalProfit = 0;
    int previous = jobs[0].finish;
    totalProfit += jobs[0].profit;
    int activity = 1;
    printf("Activity: %d, Profit: %d", activity, totalProfit);
    for(int i = 1; i < 11; i++){
        int curr = jobs[i].start;
        if(curr>=previous){
            activity = i + 1;
            totalProfit+=jobs[i].profit;
            previous = jobs[i].finish;
            printf("\nActivity: A%d, Profit: %d", activity,
jobs[i].profit);
```

```c
        }
    }
    return totalProfit;
}

int main() {
    int totalProfit;

    struct Job jobs[11] = {
        {1,4,10}, {3,5,15}, {0,6,14}, {5,7,12}, {3,9,20}, {5,9,30},
{6,10,32}, {8,11,28}, {8,12,30}, {2,14,40}, {12,16,45}
    };

    int profit = ASP(jobs);

    printf("\nTotal profit is %d", profit);

    return 0;
}
```

OUTPUT:



```
Activity: 1, Profit: 10
Activity: A4, Profit: 12
Activity: A8, Profit: 28
Activity: A11, Profit: 45
Total profit is 95
```