Name: Areeb Ansari Roll No:A3-B1-01

Practical 3

Problem Definition: Suppose you are a transport dealer and want to load a truck with

different types of boxes. Assume there are 50 types of boxes (Box-1 to Box-50), which weigh

different and that the truck has a maximum capacity (truckSize). Each box has a profit value

associated with it. It is the commission that the transporter will receive after transporting the

box. You can choose any box to put on the truck as long as the number of boxes does not

exceed truckSize. You can load partial boxes.

Tasks-1: [SUBMISSION ON CLASSROOM]

A. Load the truck using different methods: Minimum weight, Maximum profit,

Profit/weight ratio. Compute the total profit using each method and infer the best

performing method.

B. Compute the time required in each method.

Given Data:

 Capacity of truck 850 Kgs

 Weight in kg for each box:

[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]

⍰ Profit in Rs for each box:

[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]

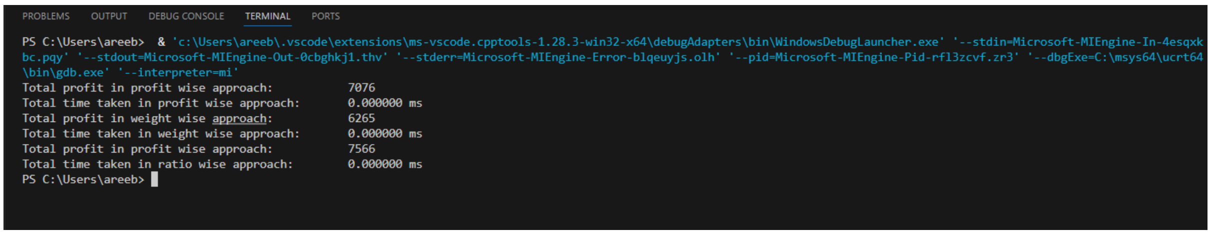
Code:

```
#include<stdio.h>

#include<limits.h>

#include <time.h>


struct Box{

   int profit;

   int weight;

   double profitPerWeight;

};


void swap(struct Box *a,struct Box *b) {

   struct Box temp = *a;

   *a = *b;

   *b = temp;
```

```
}

//Ascending
void sortByWeight(struct Box boxes[]) {
   for (int i = 0; i < 50 - 1; i++) {
      for (int j = i + 1; j < 50; j++) {
         if (boxes[i].weight > boxes[j].weight) {
            swap(&boxes[i], &boxes[j]);
         }
      }
   }
}

//Descending
void sortByProfit(struct Box boxes[]) {
   for (int i = 0; i < 50 - 1; i++) {
      for (int j = i + 1; j < 50; j++) {
         if (boxes[i].profit < boxes[j].profit) {
            swap(&boxes[i], &boxes[j]);
         }
      }
   }
}

//Descending
void sortByRatio(struct Box boxes[]) {
   for (int i = 0; i < 50 - 1; i++) {
      for (int j = i + 1; j < 50; j++) {
            if (boxes[i].profitPerWeight < boxes[j].profitPerWeight) {
                swap(&boxes[i], &boxes[j]);
            }
        }
    }
}

double loadTruck(struct Box boxes[], int capacity) {
    double totalProfit = 0.0;
    int currentWeight = 0;

    for (int i = 0; i < 50; i++) {
        if (boxes[i].weight == 0) continue;

        if (currentWeight + boxes[i].weight <= capacity) {
            totalProfit += boxes[i].profit;
            currentWeight += boxes[i].weight;
        } else {
            int remaining = capacity - currentWeight;
            totalProfit += boxes[i].profit * ((double)remaining / boxes[i].weight);
            break;
        }
    }

    return totalProfit;
}
```

```
int main(){

   clock_t start, end;
   double cpu_time_used;

   start = clock();

   int weights[50] = 
{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[50] = 
{360,83,59,130,431,67,230,52,93,125,670,892,600,38,48,147,78,256,63,17,120,164,43
2,35,92,110,22,42,50,323,514,28,87,73,78,15,26,78,210,36,85,189,274,43,33,10,19,38
9,276,312};
   struct Box boxes[50],temp[50];


   for(int i=0;i<50;i++){
      boxes[i].weight = weights[i];
      boxes[i].profit = profits[i];
      if (weights[i] == 0) {
         boxes[i].profitPerWeight = 0;
      } else {
         boxes[i].profitPerWeight = (double)profits[i] / weights[i];
      }
   }

   for (int i = 0; i < 50; i++) {
      temp[i] = boxes[i];
   }
   sortByWeight(temp);
   double profitWeight = loadTruck(temp, 850);


   for (int i = 0; i < 50; i++) {
      temp[i] = boxes[i];
   }
   sortByProfit(temp);
   double profitValue = loadTruck(temp, 850);


   for (int i = 0; i < 50; i++){
      temp[i] = boxes[i];
   }
   sortByRatio(temp);
   double profitRatio = loadTruck(temp, 850);

printf("Profit using weight sorting: %.2lf\n", profitWeight);
printf("Profit using profit sorting: %.2lf\n", profitValue);
printf("Profit using profit/weight ratio sorting: %.2lf\n", profitRatio);

end = clock();
   cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
   printf("Execution time: %f seconds\n", cpu_time_used);



   return 0;


}
```

Output:



```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\areeb>  & 'c:\Users\areeb\.vscode\extensions\ms-vscode.cpptools-1.28.3-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-4esqxk
bc.pqy' '--stdout=Microsoft-MIEngine-Out-0cbghkj1.thv' '--stderr=Microsoft-MIEngine-Error-blqeuyjs.olh' '--pid=Microsoft-MIEngine-Pid-rfl3zcvf.zr3' '--dbgExe=C:\msys64\ucrt64
\bin\gdb.exe' '--interpreter=mi'
Total profit in profit wise approach:         7076
Total time taken in profit wise approach:     0.000000 ms
Total profit in weight wise approach:         6265
Total time taken in weight wise approach:     0.000000 ms
Total profit in profit wise approach:         7566
Total time taken in ratio wise approach:      0.000000 ms
PS C:\Users\areeb> 
```

Problem Definition: In the single-machine scheduling problem, we are given a set of n

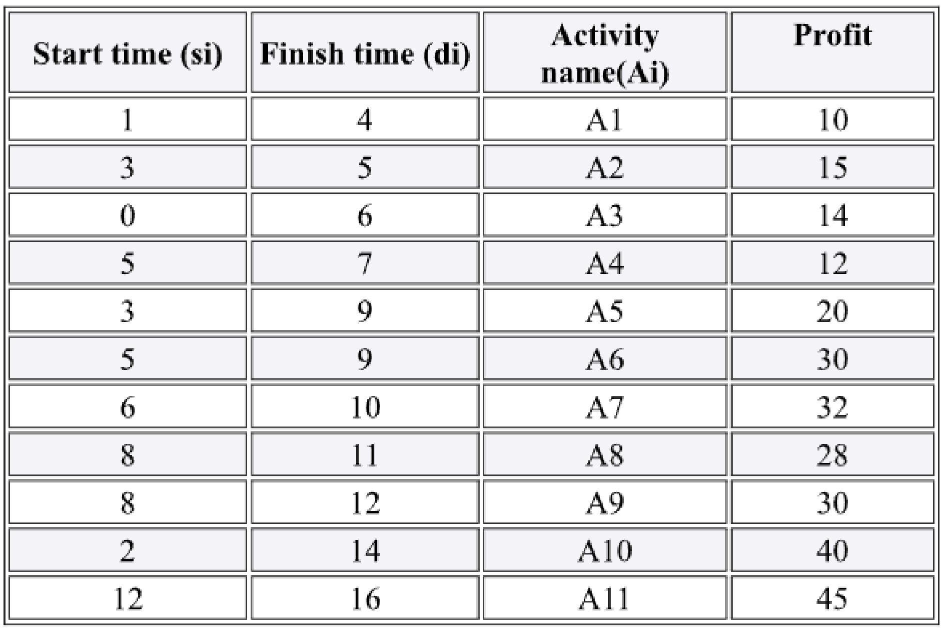
activities Ai. Each job i has a starting time si, deadline di and profit pi. At any time instant,

we can do only one job. Doing a job i earns a profit pi. Generate a solution to select the

largest set of mutually compatible jobs and calculate the total profit generated by the

machine. The greedy algorithm for single-machine scheduling selects the job using activity

selection algorithm.

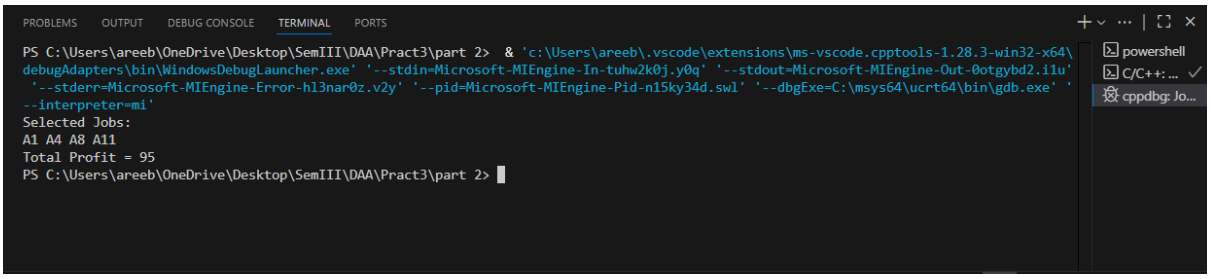


Start time (si)	Finish time (di)	Activity name(Ai)	Profit
1	4	A1	10
3	5	A2	15
0	6	A3	14
5	7	A4	12
3	9	A5	20
5	9	A6	30
6	10	A7	32
8	11	A8	28
8	12	A9	30
2	14	A10	40
12	16	A11	45

Code:

```
#include <stdio.h>

#include <time.h>


struct Activity {

    int start;

    int finish;

    int profit;

    int index;

};


void activitySelection(struct Activity activities[], int n) {

    int i, j, totalProfit = 0;

    int selected[n];

    int count = 0;


    i = 0;

    selected[count++] = i;

    totalProfit += activities[i].profit;


    for (j = 1; j < n; j++) {

        if (activities[j].start >= activities[i].finish) {

            selected[count++] = j;

            totalProfit += activities[j].profit;

            i = j;
        }
    }

    printf("Selected activities are: ");
    for (i = 0; i < count; i++) {
        printf("%d ", activities[selected[i]].index);
    }
    printf("\nTotal profit: %d\n", totalProfit);
}

int main() {
    clock_t start, end;
    double cpu_time_used;

    struct Activity activities[] = {
        {1, 4, 10, 0}, {3, 5, 15, 1}, {0, 6, 14, 2}, {5, 7, 12, 3},
        {3, 9, 20, 4}, {5, 9, 30, 5}, {6, 10, 32, 6}, {8, 11, 28, 7},
        {8, 12, 30, 8}, {2, 14, 40, 9}, {12, 16, 45, 10}
    };
    int n = sizeof(activities) / sizeof(activities[0]);

    start = clock();
    printf("Following activities are selected:\n");
    activitySelection(activities, n);
    end = clock();

    cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
    printf("Execution time: %f seconds\n", cpu_time_used);
    return 0;
}
```

Output:



```
PS C:\Users\areeb\OneDrive\Desktop\SemIII\DAA\Pract3\part 2>  & 'c:\Users\areeb\.vscode\extensions\ms-vscode.cpptools-1.28.3-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-tuhw2k0j.y0q' '--stdout=Microsoft-MIEngine-Out-0otgybd2.i1u' '--stderr=Microsoft-MIEngine-Error-hl3nar0z.v2y' '--pid=Microsoft-MIEngine-Pid-n15ky34d.swl' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Selected Jobs:
A1 A4 A8 A11
Total Profit = 95
PS C:\Users\areeb\OneDrive\Desktop\SemIII\DAA\Pract3\part 2>
```