**The Report of QUIZ 2**

**“Job Sequencing with Deadline”**

****

**Group 3**

1. Isnaini Nurul KurniaSari (05111740000010)
2. Bima Satria Ramadhan (05111740000081)
3. Meila Kamilia (05111740000189)

**Design & Analysis of Algorithms (F)**

**Institut Teknologi Sepuluh Nopember**

**Year Lessons 2018/2019**

**Job Sequencing Problem with Deadline**

* **Problem Description:**

In the industrial world, time efficiency is very important. The more efficient production process, the more total production produced or minimum costs that must be incurred. A convection company is a company that produces apparel, such as shirts, pants, shirts and jackets. Each type of clothing requires different finishing times, for example T-shirt production is faster than jacket production. The convection industry uses operators and sewing machines for each of its work stations.

In terms of product manufacturing, convection companies are included in the category of make to order (MTO) because products will be made if there are orders from customers. Every order from customers varies greatly from the type of clothing made and the amount. The purpose of this case is to solve the problem of scheduling apparel production in a convection company in order to minimize makespan (total work completion time) for each order (job order) received from customers.

Based on the order, the convection company must determine the production scheduling in the form of a sequence of types (priority) of products produced for each job and how much profit is earned for each job.

* **Points to remember**
* In this problem we have n jobs j1, j2, … jn each has an associated deadline d1, d2, … dn and profit p1, p2, ... pn.
* Profit will only be awarded or earned if the job is completed on or before the deadline.
* We assume that each job takes unit time to complete.
* The objective is to earn maximum profit when only one job can be scheduled or processed at any given time
* **Input :**

Job Deadline Profit

j1 2 60

j2 1 100

j3 3 20

j4 2 40

j5 1 20

j6 4 50

* **Output :**

Job Deadline Profit

j2 1 100

j1 2 60

j6 4 50

j4 3 40

j3 3 20

j5 1 20

dmax: 3

Required Jobs: j2 --> j1 --> j3

Max Profit: 180

* **Problem Abstraction**

The convection industry is a company that produces various kinds of apparel such as shirts, and trousers. In terms of product manufacturing, this industry uses order (MTO), meaning the product will be made if there is a customer order. Each order received varies greatly in terms of the type of clothing to be produced and the amount so that it is necessary to schedule production on each order. In preparing the production schedule, the company must be able to allocate every variety of work into the work station (sewing machine and operator) in a balanced manner to produce a minimum makespan (minimum total work completion time).

To arrange a production schedule on parallel machines to produce a minimum makespan we can use the greedy algorithm. From the results of the study, the greedy algorithm always produces the optimal solution for this case. In addition, the greedy algorithm is always the fastest in generating solutions compared to the exhaustive search algorithm. Greedy is an algorithm to find the most optimum solution which has the maximum value at each step, or it can be called a local maximum. But in most cases, greedy cannot produce optimal values, but usually can provide a solution that approaches the most optimum value.

While for sorting (sorting) our profits, we use bubble sort.

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of Ο(n2) where **n** is the number of items.

## How Bubble Sort Works?

We take an unsorted array for our example. Bubble sort takes Ο(n2) time so we're keeping it short and precise.

Bubble Sort

Bubble sort starts with very first two elements, comparing them to check which one is greater.

Bubble Sort

In this case, value 33 is greater than 14, so it is already in sorted locations. Next, we compare 33 with 27.

Bubble Sort

We find that 27 is smaller than 33 and these two values must be swapped.

Bubble Sort

The new array should look like this −

Bubble Sort

Next we compare 33 and 35. We find that both are in already sorted positions.

Bubble Sort

Then we move to the next two values, 35 and 10.

Bubble Sort

We know then that 10 is smaller 35. Hence they are not sorted.

Bubble Sort

We swap these values. We find that we have reached the end of the array. After one iteration, the array should look like this −

Bubble Sort

To be precise, we are now showing how an array should look like after each iteration. After the second iteration, it should look like this −

Bubble Sort

Notice that after each iteration, at least one value moves at the end.

Bubble Sort

And when there's no swap required, bubble sorts learns that an array is completely sorted.

Bubble Sort

Now we should look into some practical aspects of bubble sort.

* **Solution**

Job sequencing is the set of jobs, associated with the job i where deadline di >= 0 and profit pi > 0. For any job i the profit is earned if and only if the job is completed by its deadline. To complete a job, one has to process the job on a machine for one unit of time. Only one machine is available for processing the jobs.

Steps for performing job sequencing with deadline using greedy approach is as follows:

1. Sort all the jobs based on the profit in an decreasing order.
2. Let α be the maximum deadline that will define the size of array.
3. Create a solution array S with d slots.
4. Initialize the content of array S with zero.
5. Check for all jobs.
   1. If scheduling is possible a lot ith slot of array s to job i.
   2. Otherwise look for location (i-1), (i-2)...1.
   3. Schedule the job if possible else reject.
6. Return array S as the answer.
7. End.

Point to remember :

1. In this problem, we have n jobs with id jobs; j1, j2,j3,…, jn each has an associated deadline d1, d2,…,dn and profit p1,p2,…,pn.
2. Profit will only be awarded or earned if the job is completed or before the deadline.
3. We assume that each job takes a unit time to complete.
4. The objective is to earn maximum profit when only one job can be scheduled or processed at any given time.

Consider the following 5 id jobs and their associated deadline and profit:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j1 | j2 | j3 | j4 | j5 |
| Deadline | 2 | 1 | 3 | 2 | 1 |
| Profit | 60 | 100 | 20 | 40 | 20 |

Then, we can sort the id jobs according to their profit in descending order. If two or more jobs are having the same profit, we can sort them as their entry in the job list :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j2 | j1 | j4 | j3 | j5 |
| Deadline | 1 | 2 | 2 | 3 | 1 |
| Profit | 100 | 60 | 40 | 20 | 20 |

Job j3 and j5 are having profit 20 so we placed j3 first as it came before j5.

Then we find the maximum deadline value, then looking at the id jobs that we can say the max deadline value is 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j2 | j1 | j4 | j3 | j5 |
| Deadline | 1 | 2 | 2 | 3 | 1 |
| Profit | 100 | 60 | 40 | 20 | 20 |

dmax

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j2 | j1 | j4 | j3 | j5 |
| Deadline | 1 | 2 | 2 | 3 | 1 |
| Profit | 100 | 60 | 40 | 20 | 20 |

As dmax = 3, so we will have THREE slots to keep track of free time slots, and then we set the time slot status to **EMPTY**;

|  |  |  |  |
| --- | --- | --- | --- |
| Time slot | 1 | 2 | 3 |
| status | EMPTY | EMPTY | EMPTY |

Total number of job is 5 so we can write n = 5 and dmax = 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j2 | j1 | j4 | j3 | j5 |
| Deadline | 1 | 2 | 2 | 3 | 1 |
| Profit | 100 | 60 | 40 | 20 | 20 |

If we look at job j2, it has a deadline 1, it means that we have to complete job j2 in time slot 1 if we want to earn its profit.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j2 | j1 | j4 | j3 | j5 |
| Deadline | 1 | 2 | 2 | 3 | 1 |
| Profit | 100 | 60 | 40 | 20 | 20 |

**We get n = 5 and dmax = 3**

If we look at job 2, it has a deadline 1, it means that we have to complete job j2 in time slot 1 if we want to earn its profit.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j2 | j1 | j4 | j3 | j5 |
| Deadline | 1 | 2 | 2 | 3 | 1 |
| Profit | 100 | 60 | 40 | 20 | 20 |

Similarly, if we look at id job j1. It has a deadline 2, it means we have to complete job j1 on or before time slot 2 in order to earn its profit.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j2 | j1 | j4 | j3 | j5 |
| Deadline | 1 | 2 | 2 | 3 | 1 |
| Profit | 100 | 60 | 40 | 20 | 20 |

Similarly, if we look at job j3 it has a deadline 3 this means we have to complete job j3 on or before time slot 3 in order to earn its profit.

Our objective is to select jobs that will give us higher profit.

i

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j2 | j1 | j4 | j3 | j5 |
| Deadline | 1 | 2 | 2 | 3 | 1 |
| Profit | 100 | 60 | 40 | 20 | 20 |

I = 1

k = min(dmax, Deadline(i) )

k = min(3, Deadline(1))

k = min (3,1)

k = 1

is k >= 1?

1 >= 1

YES it’s true

Then we can check to the time slot(k) == EMPTY

|  |  |  |  |
| --- | --- | --- | --- |
| Time slot | 1 | 2 | 3 |
| status | EMPTY | EMPTY | EMPTY |

Time slot(1) == EMPTY

Because Time slot(1) is EMPTY, we can fill the time slot (1) with j2

|  |  |  |  |
| --- | --- | --- | --- |
| Time slot | 1 | 2 | 3 |
| status | j2 | EMPTY | EMPTY |

i

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j2 | j1 | j4 | j3 | j5 |
| Deadline | 1 | 2 | 2 | 3 | 1 |
| Profit | 100 | 60 | 40 | 20 | 20 |

I = 2

K = min(dmax, Deadline(i) )

K = min(3, Deadline(2))

K = min (3,2)

K = 2

is k > = 1 ?

2 > = 1 ; Yes, it’s true

After that we can check the time slot(k) == EMPTY. Then we get the time slot (2) == EMPTY

|  |  |  |  |
| --- | --- | --- | --- |
| Time slot | 1 | 2 | 3 |
| Status | j2 | EMPTY | EMPTY |

YES, time slot(2) is EMPTY

Because time slot(2) is empty, we can fill the time slot (2) with j1

|  |  |  |  |
| --- | --- | --- | --- |
| Time slot | 1 | 2 | 3 |
| status | j2 | j1 | EMPTY |

i

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j2 | j1 | j4 | j3 | j5 |
| Deadline | 1 | 2 | 2 | 3 | 1 |
| Profit | 100 | 60 | 40 | 20 | 20 |

We can assume that :

n = 5

dmax = 3

i = 3

k = min (dmax, deadline(i))

k = min (3, deadline (3) )

k = min (3,2)

k = 2

is k > = 1 ?

2 > = 1

**YES**

After that we can check the time slot(k) == EMPTY or NOT EMPTY

|  |  |  |  |
| --- | --- | --- | --- |
| Time slot | 1 | 2 | 3 |
| status | j2 | j1 | EMPTY |

Time slot (2) is **NOT EMPTY**

Because time slot (k) is NOT EMPTY, so we reduce k by 1

n = 5

k = 1

is k > = 1 ?

1 > = 1

YES **it’s true**

Then, check time slot(k) == EMPTY

|  |  |  |  |
| --- | --- | --- | --- |
| Time slot | 1 | 2 | 3 |
| status | j2 | j1 | EMPTY |

Time slot (1) is **NOT EMPTY**

Because time slot (k) is NOT EMPTY, so we reduce k by 1

i

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j2 | j1 | j4 | j3 | j5 |
| Deadline | 1 | 2 | 2 | 3 | 1 |
| Profit | 100 | 60 | 40 | 20 | 20 |

With n = 5; dmax = 3

We can assume that:

i = 4

k = min (dmax , deadline (i) )

k = min (3, deadline (4) )

k = min (3,3)

k = 3

is k >= 1?

so, 3 >= 1

**YES, it’s TRUE**

|  |  |  |  |
| --- | --- | --- | --- |
| Time slot | 1 | 2 | 3 |
| Status | j2 | j1 | EMPTY |

Time slot (3) is  **EMPTY**

Because time slot(3) is EMPTY, we can fill the time slot (3) with j3

|  |  |  |  |
| --- | --- | --- | --- |
| Time slot | 1 | 2 | 3 |
| status | j2 | j1 | j3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| ID Job | j2 | j1 | j4 | j3 | j5 |
| Deadline | 1 | 2 | 2 | 3 | 1 |
| Profit | 100 | 60 | 40 | 20 | 20 |

n = 5

dmax = 3

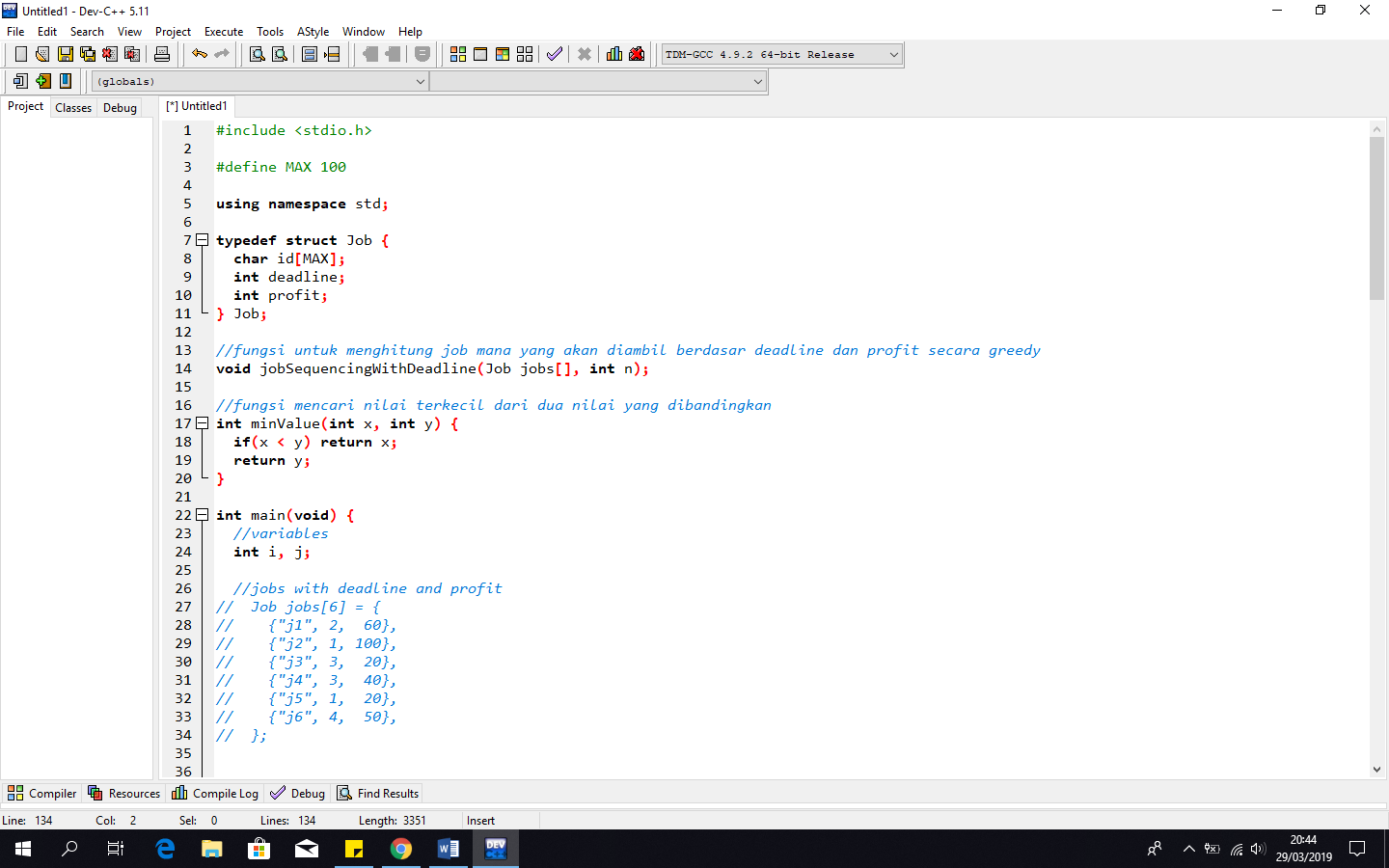
required job sequence is

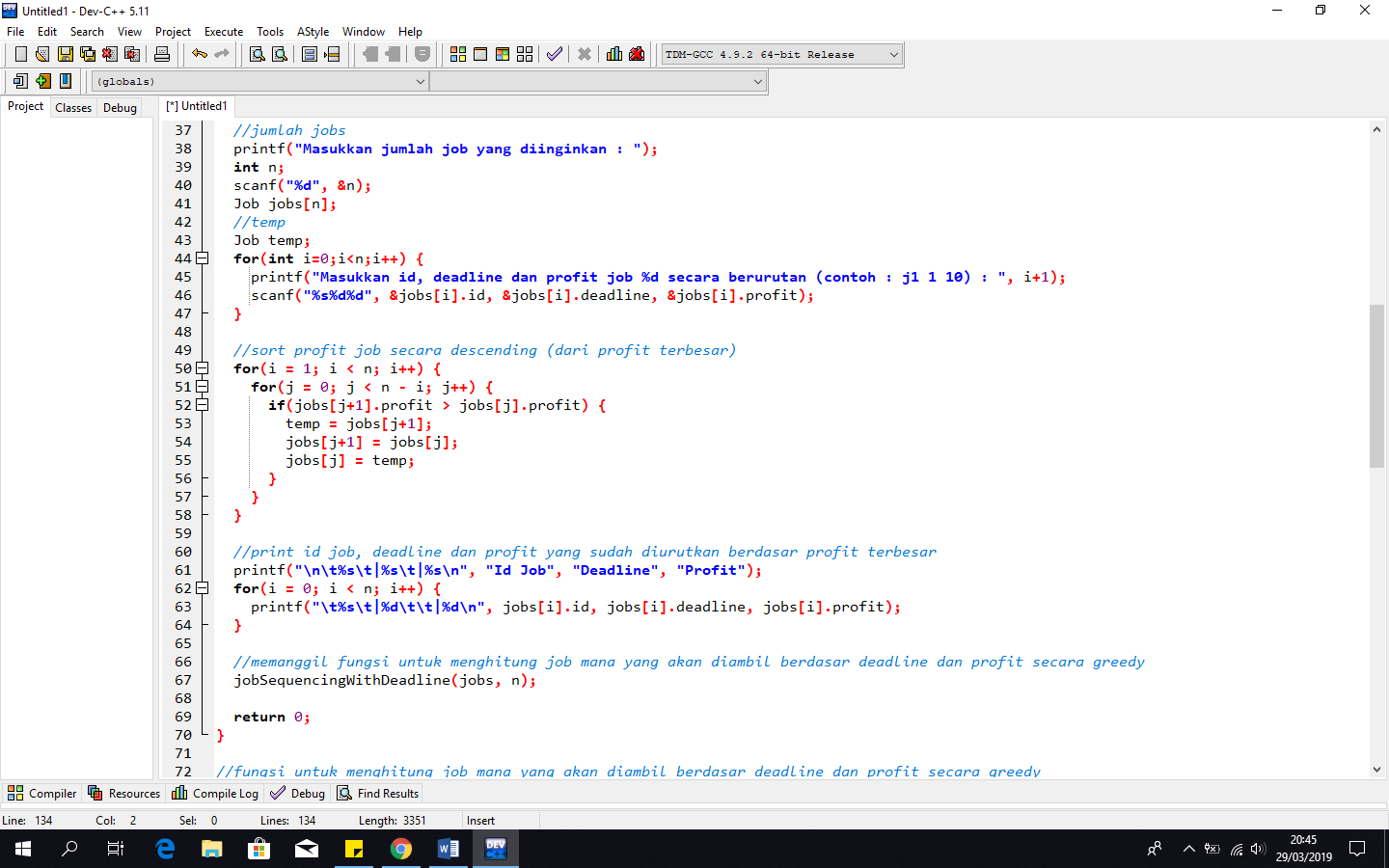
j2 🡪 j1 🡪 j3

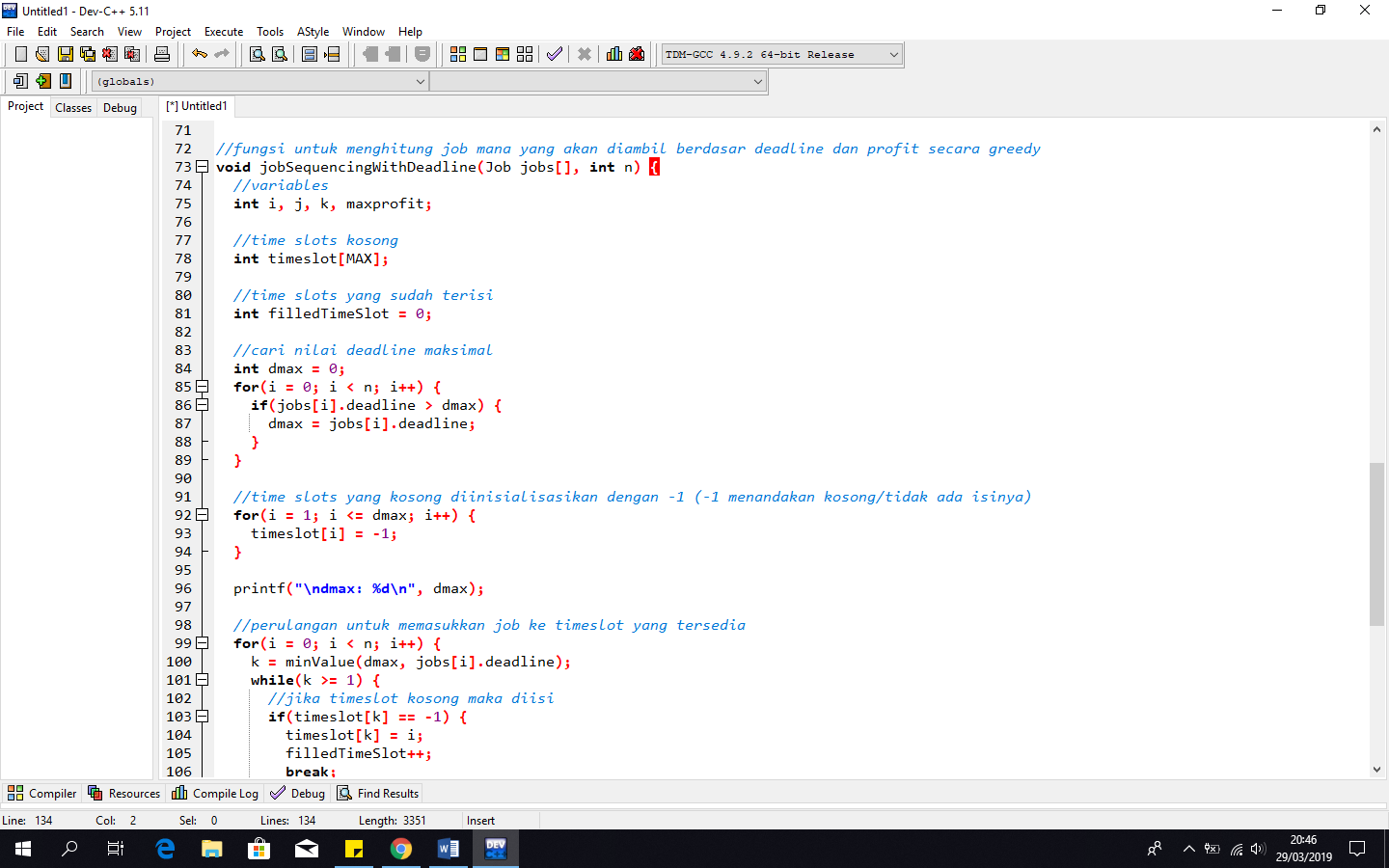
and we know that max profit is

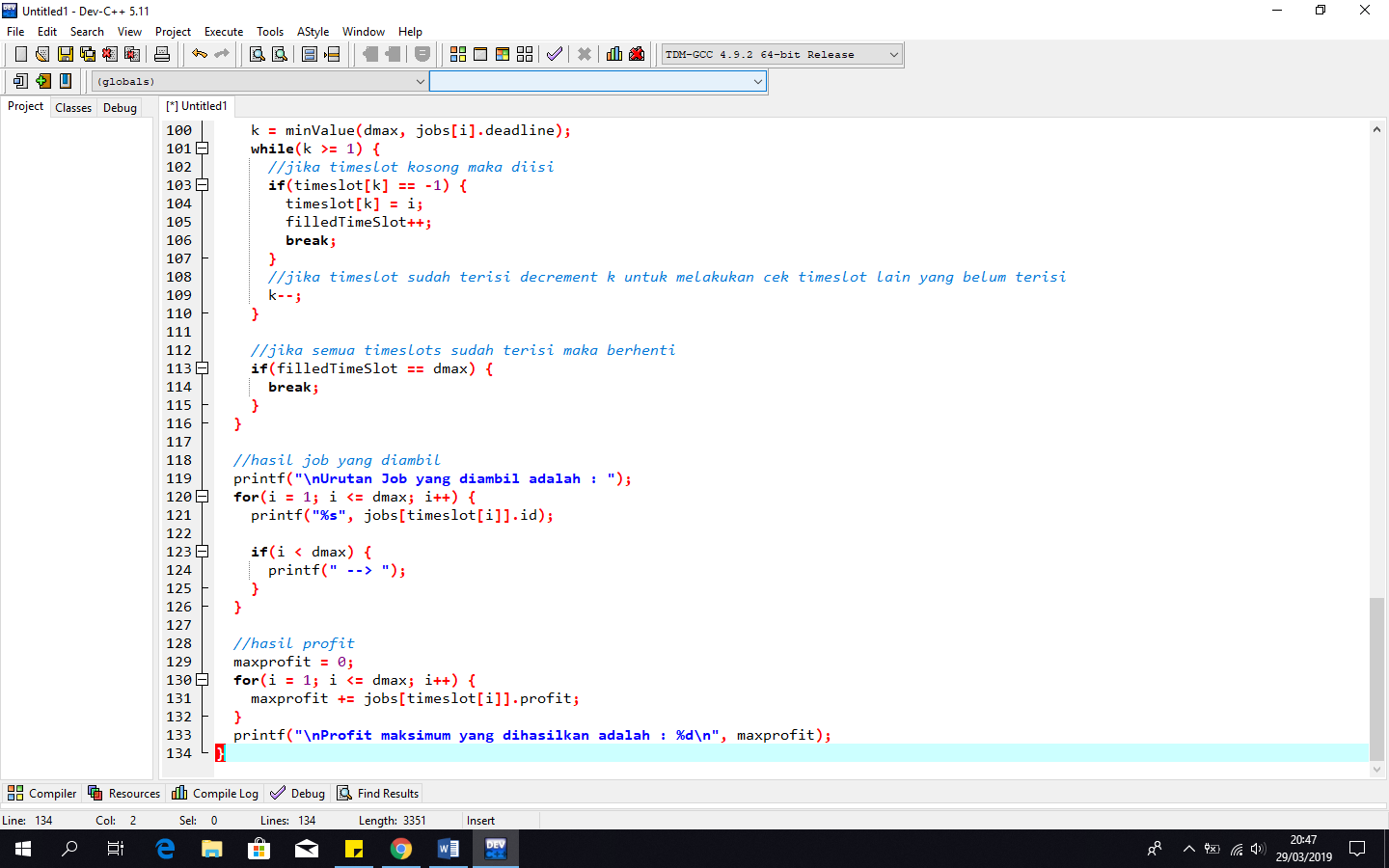
100 + 60 +20 = 180

* Source Code









* Output Program

