Benedictus Alvian Sofjan
3035603095

Question 1

./tesla_factory.out 3 20 1 2 3

```
Name: Benedictus Alvian Sofjan   UID: 3035603095
Production goal: 3, num space: 20, num typeA: 1, num typeB: 2, num typeC: 3
====Final Report====
Num free space: 20
Produced Skeleton: 0
Produced Engine: 0
Produced Chassis: 0
Produced Body: 0
Produced Window: 0
Produced Tire: 0
Produced Battery: 0
Produced Car: 3
Production task completed! 3 cars produced.
Production time: 35.001254
```

How many units of storage space can be considered as sufficient for sure?

Given a sufficiently bad algorithm, no units of storage space can be considered sufficient. Even to produce 100 cars with 100 spaces with 100 robots, if all 100 robots decide to acquire space to assemble the body, they would all be stuck in a deadlock. To make one car, we need 16 units of storage space to be certain no deadlock would occur given all possible algorithms. If we are talking about the minimum number of units of storage space needed, that would be two, but all cars would then have to created sequentially.

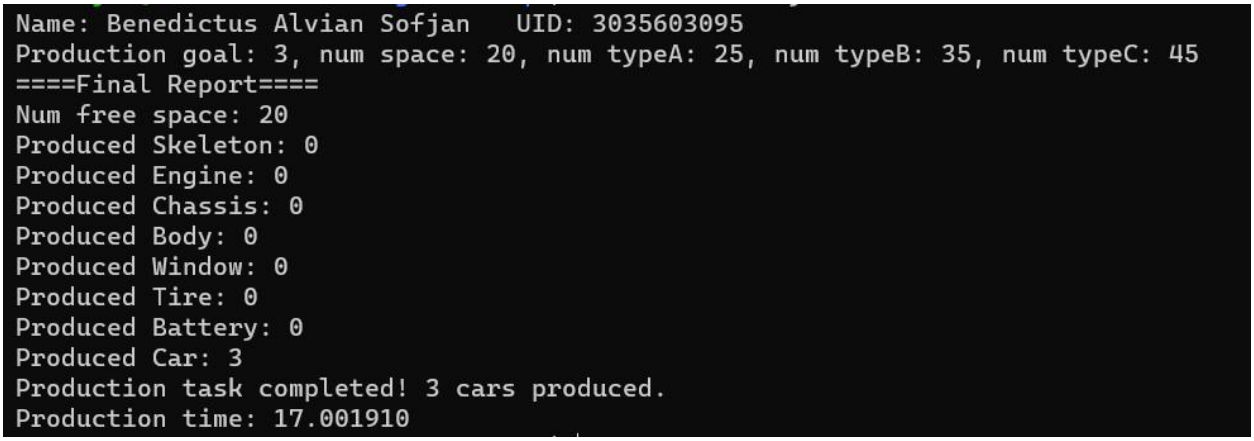
Is it always true that as the number of robot increases, the production time decreases?

./tesla_factory.out 3 20 2 3 4 results in a substantial decrease of time.

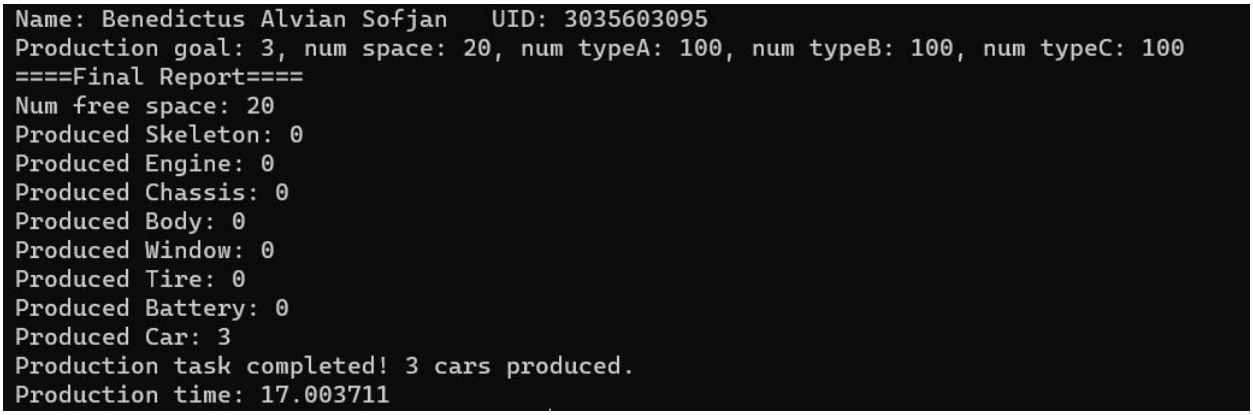
```
Name: Benedictus Alvian Sofjan   UID: 3035603095
Production goal: 3, num space: 20, num typeA: 2, num typeB: 3, num typeC: 4
====Final Report====
Num free space: 20
Produced Skeleton: 0
Produced Engine: 0
Produced Chassis: 0
Produced Body: 0
Produced Window: 0
Produced Tire: 0
Produced Battery: 0
Produced Car: 3
Production task completed! 3 cars produced.
Production time: 21.000824
```

Now let us increase the number of robots even more.

./tesla_factory.out 3 20 25 35 45



./tesla_factory.out 3 20 100 100 100



As apparent above, as the number of robot increases, the production time does not necessarily decrease. As each robot takes different time to make different parts, it is not necessarily true more robots would always decrease production time. At a certain limit as well, the production time plateaus as the number of robots increases due to maximum parallelization achieved.

How can the number of cars and the number of robots of each type affect the production time (open-ended question, share your thoughts)?

The production time is affected by a multitude of factors. Factors include the type of robots, the number of cars, the algorithm employed, the number of free space and current CPU resource. More cars would naturally mean longer production time, assuming constant value for every other variable. The number of robots increased would also lead us to naturally conclude that more robots mean less production time, but as we have seen in the previous example, the number of robots at one point will not further

decrease production time as its use has saturated. The increased number of robots will not necessarily decrease production time, if there is less robot of one type and more of the other, as they have different processing time for different parts, and given the algorithm, it might actually conversely increase the production time.

Question 2

I changed the scheduling of the tasks (body -> body parts -> car -> car parts) to ensure no deadlock occurs and to enforce this order of creation, I put a mutex lock on struct Task_t to make it global for all robots and locked the mutex before dequeue the task queue and unlocked it after the thread has entered the simpleWork() function. I initialized the mutex lock in the beginning of main() and destroyed it right before main() exits.

Question 3

How close your program can get to the theoretical shortest production time with constraints on the number of robots and storage space?

My program performs well compared to other optimized algorithms given a large number of storage space and robots to manage with the storage space. When the number of storage space is very limited and there are many robots (e.g. when not a lot of parallelization can be achieved), my program underperforms compared to other algorithms that matches each job with the optimal robot to perform it on. As the number of robots and storage space increase, my program gets closer to the theoretical shortest production time. In situation of severe lack of storage space, my program underperforms.

Briefly introduce your optimized/improved algorithm in the report.

I have kept the same algorithm with Question 2 as I believe this is already optimized. I thought of capping the number of robots in Question 2 to the number of storage space available, as only one robot can work on one unit of storage space (and this also prevents deadlock) and then to make the number of active robot threads in Question 3 to be unlimited regardless of the number of space available. But I have done that in Question 2, so I will keep it as is.

How each factor (number of cars, number of storage space and number of robots) can affect your production time?

As the number of storage space increases linearly with the number of robots, the production time per car decreases exponentially fast as optimal parallelization can be reached. But after hitting 20 units of storage space, the production time decreases gradually and eventually plateaus, even as we set the storage space to 50 or 60 and increase the number of robots as well.

The figure below shows that as the number of robots increases with fixed number of storage space and production goals, the production time decreases.

```
basofjan@workbench:~/Assignment2/q2$ ./tesla_factory.out 2 10 2 2 2
Name: Benedictus Alvian Sofjan   UID: 3035603095
Production goal: 2, num space: 10, num typeA: 2, num typeB: 2, num typeC: 2
====Final Report====
Num free space: 10
Produced Skeleton: 0
Produced Engine: 0
Produced Chassis: 0
Produced Body: 0
Produced Window: 0
Produced Tire: 0
Produced Battery: 0
Produced Car: 2
Production task completed! 2 cars produced.
Production time: 24.001623
basofjan@workbench:~/Assignment2/q2$ ./tesla_factory.out 2 10 4 4 4
Name: Benedictus Alvian Sofjan   UID: 3035603095
Production goal: 2, num space: 10, num typeA: 4, num typeB: 4, num typeC: 4
====Final Report====
Num free space: 10
Produced Skeleton: 0
Produced Engine: 0
Produced Chassis: 0
Produced Body: 0
Produced Window: 0
Produced Tire: 0
Produced Battery: 0
Produced Car: 2
Production task completed! 2 cars produced.
Production time: 18.000929
```

The figure below shows us that as the number of storage space increases with other variables as constant (i.e. production goal = 2 and robots = 50, 50, 50), the production time per car decreases by a lesser amount for every unit space.

```
basofjan@workbench:~/Assignment2/q2$ ./tesla_factory.out 2 5 50 50 50
Name: Benedictus Alvian Sofjan   UID: 3035603095
Production goal: 2, num space: 5, num typeA: 50, num typeB: 50, num typeC: 50
====Final Report====
Num free space: 5
Produced Skeleton: 0
Produced Engine: 0
Produced Chassis: 0
Produced Body: 0
Produced Window: 0
Produced Tire: 0
Produced Battery: 0
Produced Car: 2
Production task completed! 2 cars produced.
Production time: 22.002551
basofjan@workbench:~/Assignment2/q2$ ./tesla_factory.out 2 10 50 50 50
Name: Benedictus Alvian Sofjan   UID: 3035603095
Production goal: 2, num space: 10, num typeA: 50, num typeB: 50, num typeC: 50
====Final Report====
Num free space: 10
Produced Skeleton: 0
Produced Engine: 0
Produced Chassis: 0
Produced Body: 0
Produced Window: 0
Produced Tire: 0
Produced Battery: 0
Produced Car: 2
Production task completed! 2 cars produced.
Production time: 18.002061
basofjan@workbench:~/Assignment2/q2$ ./tesla_factory.out 2 20 50 50 50
Name: Benedictus Alvian Sofjan   UID: 3035603095
Production goal: 2, num space: 20, num typeA: 50, num typeB: 50, num typeC: 50
====Final Report====
Num free space: 20
Produced Skeleton: 0
Produced Engine: 0
Produced Chassis: 0
Produced Body: 0
Produced Window: 0
Produced Tire: 0
Produced Battery: 0
Produced Car: 2
Production task completed! 2 cars produced.
Production time: 15.001882
```

The production time converges roughly around units of storage space = 20 (assuming the number of each type of robots stay the same and does not bottleneck the number of storage space). In the case of production goal = 2 and num TypeA = num TypeB = Num TypeC = 50, the production time converges almost precisely around 15 seconds.

```
basofjan@workbench:~/Assignment2/q2$ ./tesla_factory.out 2 20 50 50 50
Name: Benedictus Alvian Sofjan   UID: 3035603095
Production goal: 2, num space: 20, num typeA: 50, num typeB: 50, num typeC: 50
====Final Report====
Num free space: 20
Produced Skeleton: 0
Produced Engine: 0
Produced Chassis: 0
Produced Body: 0
Produced Window: 0
Produced Tire: 0
Produced Battery: 0
Produced Car: 2
Production task completed! 2 cars produced.
Production time: 15.002467
basofjan@workbench:~/Assignment2/q2$ ./tesla_factory.out 2 40 50 50 50
Name: Benedictus Alvian Sofjan   UID: 3035603095
Production goal: 2, num space: 40, num typeA: 50, num typeB: 50, num typeC: 50
====Final Report====
Num free space: 40
Produced Skeleton: 0
Produced Engine: 0
Produced Chassis: 0
Produced Body: 0
Produced Window: 0
Produced Tire: 0
Produced Battery: 0
Produced Car: 2
Production task completed! 2 cars produced.
Production time: 15.002195
```

<u>When does the number of storage space start to affect the production time?</u>

When the number of storage space is increased from 2 to 3, the greatest reduction in production time is observed.

<u>What is the theoretical best performance for given number of robots, number of storage space and number of cars to be made?</u>

For the production goal = 1 and assuming no space and robot bottleneck, the theoretical best performance is 10 seconds. For production goal = 2, assuming no space and robot bottleneck, the theoretical best performance is also 10 seconds, as the robots could work on the second car simultaneously as well.

For my program of production goal = 2, it seems my theoretical limit is 15 seconds, as evident above.

<u>What cause the performance difference between the theoretical best and your implementation?</u>

Inefficient allocation of tasks. The tasks are randomly allotted to any robot, and not always the optimal robot, so my implementation will result in a production time averaged amongst the type of robots and their proportions to one another. If all robots are of type C, then the production time is averaged to type C. If there is equal proportion amongst A, B, and C, then production time is the average time needed for the three of them working concurrently to produce a car. If there is 90% robot A and 10% robot B, then the production time is heavily skewed towards those of type A.

Given the number of robots, number of storage space, and number of cars, can you predict the production time?

Given an extraneous number of robots and storage space, my algorithm can always produce any number of cars in 15 seconds as shown below.

```
basofjan@workbench:~/Assignment2/q2$ ./tesla_factory.out 6 300 50 50 50
Name: Benedictus Alvian Sofjan   UID: 3035603095
Production goal: 6, num space: 300, num typeA: 50, num typeB: 50, num typeC: 50
====Final Report====
Num free space: 300
Produced Skeleton: 0
Produced Engine: 0
Produced Chassis: 0
Produced Body: 0
Produced Window: 0
Produced Tire: 0
Produced Battery: 0
Produced Car: 6
Production task completed! 6 cars produced.
Production time: 15.003228
basofjan@workbench:~/Assignment2/q2$ ./tesla_factory.out 10 300 80 80 80
Name: Benedictus Alvian Sofjan   UID: 3035603095
Production goal: 10, num space: 300, num typeA: 80, num typeB: 80, num typeC: 80
====Final Report====
Num free space: 300
Produced Skeleton: 0
Produced Engine: 0
Produced Chassis: 0
Produced Body: 0
Produced Window: 0
Produced Tire: 0
Produced Battery: 0
Produced Car: 10
Production task completed! 10 cars produced.
Production time: 15.005160
```

Which part of your program can be further optimized to shorten the production time?

I could have implemented the allocation of jobs to the most optimal robot that is currently free. This will result in a performance increase much closer to the theoretical limit. To even have a more aggressive algorithm, given sufficient storage space, it is not necessary to reserve space (and robots) for building the body and car part first in order to shave off a few seconds and achieve more parallelization in that brief span of time. This would increase performance by a bit. But there is a significant risk of deadlock if you have not done your Math.

Can your scheduling algorithm be applied to somewhere else (other applications)?

This algorithm might be used in a real assembly factory. To do that, we need to cap the number of working robots to the number of storage space as the cost for switching robots to the same space is high in real life.