

Nama : Fadhil Dzikri Aqila
NIM : 1103213136
Kelas : TK-45-G09

Prerequisites Exam The Construct

1. Isi kode task1.sh

The screenshot shows the Construct AI interface. On the left, the 'Task 1' instructions are displayed. The main area shows a file explorer with a directory structure and a code editor with the contents of 'task1.sh'. The bottom status bar indicates 'Prerequisites Exam' is 100% complete.

Task 1

Inside the **linux_exam** folder, create a new bash script named **task1.sh**, that does the following:

- First, it moves inside the **linux_exam** folder.
- Once it is there, it generates a folder structure like the following one: `this->is->my->linux->exam` (check *Specifications*)
- Inside the final folder, named **exam**, it creates a new file named **my_file.py**
- Finally, it prints to the screen the following string:

```
In [ ]:
```

```
This bash script has finished!
```

Specifications

- The string printed at the end **MUST** be exactly the same as the one showed above.
- The full folders structure has to be like this:

```
In [ ]:
```

```
/home/user/catkin_ws/src/linux_exam/this/is/
```

Task 2

Given the following ROS commands:

To make the Turtlebot robot perform a **small square** movement:

```
In [ ]:
```

```
roslaunch linux_exam small_square.py
```

To make the Turtlebot robot perform a **medium square** movement:

```
In [ ]:
```

```
roslaunch linux_exam medium_square.py
```

task1.sh

```
1 #!/bin/bash
2 mkdir -p this/is/my/linux/exam
3 cd /home/user/catkin_ws/src/linux_exam/this/is/my/li
4 touch my_file.py
5 echo "This bash script has finished!"
```

2. Isi kode task2.sh

The screenshot shows the Construct AI interface. On the left, the 'Task 2' instructions are displayed. The main area shows a file explorer with a directory structure and a code editor with the contents of 'task2.sh'. The bottom status bar indicates 'Prerequisites Exam' is 100% complete.

Task 2

Given the following ROS commands:

To make the Turtlebot robot perform a **small square** movement:

```
In [ ]:
```

```
roslaunch linux_exam small_square.py
```

To make the Turtlebot robot perform a **medium square** movement:

```
In [ ]:
```

```
roslaunch linux_exam medium_square.py
```

task2.sh

```
1 #!/bin/bash
2 if [ $1 == 'small_square' ]; then
3   roslaunch linux_exam small_square.py
4
5 elif [ $1 == 'medium_square' ]; then
6   roslaunch linux_exam medium_square.py
7
8 elif [ $1 == 'big_square' ]; then
9   roslaunch linux_exam big_square.py
10
```

3. Isi kode task3.sh

Task 3

Inside the **linux_exam** folder, create a new bash script, named **task3.sh**, that does the following:

- First, it goes to the folder named **exam**, which you created in Task 1.
- Once there, it removes any existing file, and it creates 3 new ones, named like this: **exam1.py**, **exam2.py** and **exam3.py**.
- Finally, it assigns to each file the following permissions:

- exam1.py:**
 - Owner: Read, Write and Execute
 - Group: Read and Execute
 - All others: Read
- exam2.py:**
 - Owner: Read and Execute
 - Group: None
 - All others: Execute
- exam3.py:**
 - Owner: Write
 - Group: Read
 - All others: Execute

Part 2: Python for Robotics

Task 0

The screenshot shows a web browser at <https://app.theconstruct.ai/Desktop>. The left sidebar displays the task instructions. The main editor shows the **task3.sh** script with the following content:

```
1 #!/bin/bash
2 cd ~/catkin_ws/src/linux_exam/this/is/my/linux/exam
3 rm -rf *
4 touch exam1.py exam2.py exam3.py
5 chmod 754 exam1.py
6 chmod 501 exam2.py
7 chmod 241 exam3.py
```

The right sidebar shows a 3D simulation of a robot in a square arena. The bottom status bar indicates the 'Prerequisites Exam' is 100% complete.

4. Isi kode task1.py

Task 1

Inside the **python_exam** folder, create a new Python script named **task1.py**. Inside this script, create a function named **get_highest_lowest**. This function does the following:

- First, it creates an instance of the **RobotControl** class.
- Second, it gets all the values of the laser readings and it stores them into a list.
- Then, it compares all these laser values that you have stored in the list in order to detect the highest value and the lowest one.
- Finally, it returns the **position in the list** of the highest and lowest values (check Specifications).

Specifications

- The name of the function **MUST BE** the one specified above: **get_highest_lowest**
- Bear in mind that the function doesn't have to return the values (highest and lowest), but the **position in the list** of those values.
- Your final (after you have tested that it works properly) program **MUST** contain the **get_highest_lowest** function, but **MUST NOT** contain any call to this function.
- The function has to return the position values in an **specific order**: first, it returns the position of the

The screenshot shows the same web browser interface. The left sidebar displays the task instructions. The main editor shows the **task1.py** script with the following content:

```
1 from robot_control_class import RobotControl
2 import math
3
4 def get_highest_lowest():
5     rc = RobotControl()
6     laser = rc.get_laser_full()
7     laser_dict = {}
8     output = []
9     for i, elem in enumerate(laser):
10         if not math.isinf(elem):
11             laser_dict[i] = elem
12     max_val = max(laser_dict.values())
13     min_val = min(laser_dict.values())
14     print(max_val, min_val)
```

The right sidebar shows the same 3D simulation of a robot in a square arena. The bottom status bar indicates the 'Prerequisites Exam' is 100% complete.

5. Isi kode task2.py

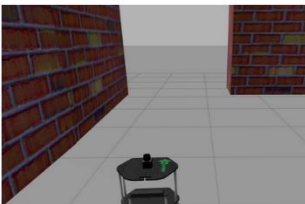
Task 2

Inside the `python_exam` folder, create a new Python script, named `task2.py`, that does the following:

- First, it starts moving the robot forwards while it captures the laser readings in front of the robot.
- When the laser readings detect that there's an obstacle (the wall) at less than 1 meter in front of the robot, the robot will stop its movement.
- After it stops, the robot will turn 90 degrees to his right, facing the opening corner in the room (check *Specifications*).

Specifications

- After turning 90 degrees, the robot **MUST** end in the orientation showed below (facing the opening corner):



```
catkin_ws > src > python_exam > task2.py > ...
1 from robot_control_class import RobotControl
2 import math
3
4 rc = RobotControl()
5 laser = rc.get_laser_full()
6
7 while laser[360] >= 1:
8     rc.move_straight()
9     laser = rc.get_laser_full()
10 rc.stop_robot()
11
12 rc.turn("clockwise", 0.2, 7)
```

Prerequisites Exam 100%

6. Isi kode task3.py

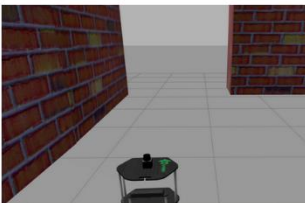
Task 2

Inside the `python_exam` folder, create a new Python script, named `task2.py`, that does the following:

- First, it starts moving the robot forwards while it captures the laser readings in front of the robot.
- When the laser readings detect that there's an obstacle (the wall) at less than 1 meter in front of the robot, the robot will stop its movement.
- After it stops, the robot will turn 90 degrees to his right, facing the opening corner in the room (check *Specifications*).

Specifications

- After turning 90 degrees, the robot **MUST** end in the orientation showed below (facing the opening corner):




```
catkin_ws > src > python_exam > task3.py > ExamControl > ...
1 from robot_control_class import RobotControl
2 import math
3
4 class ExamControl:
5     def get_laser_readings(self):
6         output = []
7         rc = RobotControl()
8         laser = rc.get_laser_full()
9         output.append(laser[719])
10        output.append(laser[0])
11        return output
12
13    def main(self):
14        rc = RobotControl()
15        d_left, d_right = self.get_laser_readings()
16
```

Prerequisites Exam 100%

7. Nilai akhir Prerequisites Exam The Construct

Course simulation - Prerequisite: X

https://app.theconstruct.ai/Desktop



Gradebot

Hi there! Here to check that your assessment package works as expected. I award marks and provide helpful feedback while at it so you can take action where needed.

Please note: I'm not even human, let alone perfect. I could use some feedback too!

If you think your code works correctly but I'm failing you, kindly file a dispute using the help button (bottom-right corner of page), as soon as possible, ideally after the 2nd or 3rd attempt.


-> Your current score is 9.0/10.0.

-> Are you sure you want to submit your exam for grading again? If you do, the score will be overwritten.

-> You have submitted 7 time(s) and you have infinite trial(s) left (maximum allowed trials: infinite).

SUBMIT FOR GRADING

CLOSE



16:59

09/10/2024