RECRUITMENT SCREENING TEST

SOFTWARE MODULE

INSTRUCTIONS

- **This screening test consists of two sections, A and B.**
    - Section A tests you on ROS
    - Section B tests you on other vital aspects.
    - There are three BONUS questions (Optional) at the end of the application. Attempting them will reflect positively on you.

- **Try to attempt as many questions as possible in both sections. Please do note that even if you can't get a question right, do explain your approach and thought process.**

- **Submission:**
    - Submit to the form: https://forms.gle/ZgwC7XRfb45kH4mdA
    - Format: a Public GitHub repository containing all the required files and a README.

- **For the bonus questions, add the required files (pdf/png/other)in the repo itself and mention the submissions clearly in the README.**
- **The code must be clear, concise, and well commented.**
- **Any kind of plagiarism will be dealt with extremely strictly. Acknowledge each and every resource used.**

- **For any queries and clarifications contact :**
    - Dhruv Maroo (CS20B025) - 8141231731
    - Dhanush A (CH20B036) - 9840722578

The README should be in the following format

Submission for Team Abhiyaan

=====================================

(Example)

Name:
Thor Odinson

Roll no:
XX20B010

Previous Experience

Killed Thanos using Stormbreaker

Current PORs:

Prince of Asgard
God of Thunder
Member of The Avengers

Why I want to work in the team:

Because I think the team is the best

Relevant Courses:

In Institute

CS110
0
ME220
W
NA525
1

0 - Passed
W - Attendance Shortage
1 - Doing

Online

CS20SI
Andrew NG Deeplearning.ai
David Silver RL

Other Relevant Things:

I am the one.

Did you attempt bonus questions:

1. Yes
2. Yes
3. No

All the very best. Hope to see you in our team soon!

SECTION - A:

This section is the most important section. This section tests you on your ability to work with the ROS framework.

ROBOT OPERATING SYSTEM (ROS)

Welcome to the fascinating world of ROS!

ROS is the key to everything that we do at Abhiyaan. Hence it's imperative that you focus on mastering the basics of this framework.

We suggest that you definitely work through the: ROS Tutorials which is your encyclopedia for anything and everything ROS. ROS Wiki is one of the most helpful resources out there. You can also refer to the book **Programming Robots with ROS**, *O'Reilly Publications*, *Morgan Quigley*, *Brian Gerkey* and *William Smart*.

TASK 1

- Setup Ubuntu 20.04 on your computer. You will need to dual boot Linux or install a virtual machine. WSL is discouraged because it causes multiple issues.

- Install ROS Noetic in Ubuntu 20.04.

TASK 2

- Create a catkin workspace after installing **ROS Noetic**. Do the following subtasks inside the workspace.

- Write a subscriber node and a publisher node in the same package and launch both using a launch file.

  Node1 : To publish the string "Team Abhiyaan rocks:" to the
  Topic: /team_abhiyaan
  Datatype: String

  Node2 : To subscribe to the above publisher and print the message on the console

  Write a single launch file to run both nodes.

- Now write a node which subscribes to the above publisher and reverses each word in the string individually and publishes onto the

  Topic: **/naayihba_maet**
  Datatype: String

  i.e if “Team Abhiyaan Rocks” is being published to **/team_abhiyaan**, you need to publish “maeT naayihbA skcoR” to **/naayihba_maet.**

Note: The above node alone carries a small bonus if written in C++. We need to work with Python, C++, and sometimes even C in our work. Basic knowledge of multiple languages is important.

These set of tasks would have helped you understand the power of ROS - how it allows communication between nodes and how well it integrates with different programming languages.

Now that you have got a grip of the framework, let’s move on to something more interesting.

TASK 3

Before going to the next question, you should check the documentation of turtlesim in ROS. **(Note of caution: Turtles in turtlesim are much much faster than the turtles inside insti XD)**

- Open turtlesim. You will be welcomed by a turtle named ‘turtle1’ spawned at the center of the simulator. Kill this turtle using :

```
rosservice call /kill “turtle1”
```

- Spawn two turtles named “turtle1” and “turtle2” at any two positions and orientations of your choice. For example, the following command spaws “turtle1” at (1,5) facing 90 degrees wrt horizontal

```
rosservice call /spawn 1 5 1.5707 “turtle1”
```

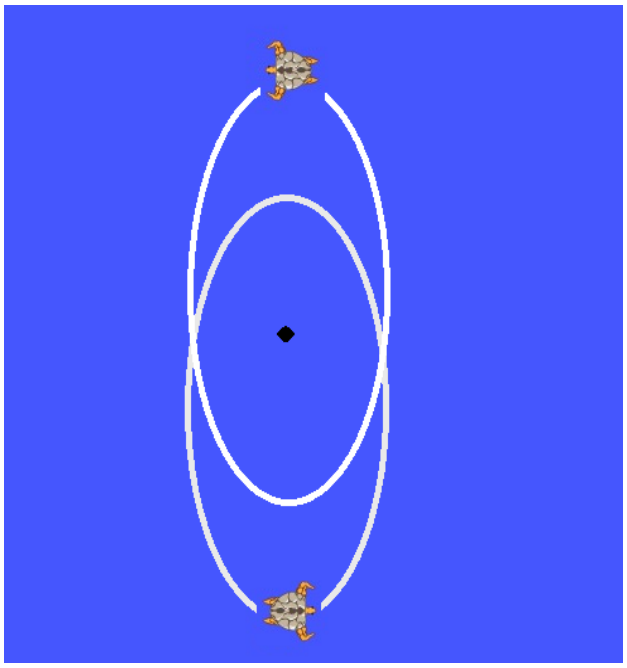
The command rosservice comes under a separate feature of ROS under services & clients. You can read up on it.

- Your task is to write a ROS Node using C++/Python to execute the following scenario:

Simulate the classical two body problem considering the two turtles to be rigid bodies. The two body problem :

Predict the motion of two objects which are abstractly viewed as point particles. The problem assumes that the two objects interact only with one another; the only force affecting each object arises from the other one.

i.e given initial positions, orientations, velocity of the two turtles (all of your choice), simulate the motion of the two turtles assuming the only force acting on them is the gravitational force F = G*m1*m2/(r^2) due to the other turtle.



Feel free to choose convenient values for G,m1 and m2. (G can be any value such as 1,2,10, etc...)

The real crux of the problem is choosing the right parameters as well as the initial velocities of the turtles in a way such that the problem is well-formulated. i.e to obtain a path that is visually appealing as well.

Note : The above figure is not a true representation of the task. It is not up to scale as well. It is just to give you a rough idea.

Hints:
Try exploring the outputs of the following while turtlesim is running.

*** rostopic list** *** rosmsg show** *** rostopic info**

Try to subscribe to the pose of both turtles and use them to perform the simulation. *Make any required assumptions and clearly state the* *assumptions you have made.*

SECTION - B:

This section evaluates your interest in the sub-topics. We suggest you try out everything! Even if you do not know anything, you will enjoy the learning experience!

SUBSECTION B.1: PROGRAMMING

1. You are given a 2D matrix of dimensions m * n consisting of integers, and an integer k. Your task is to print “true” if the element k lies in the matrix and “false” otherwise. If the element exists then print the index in a new line.
Each row and column is in sorted order, and the first element of a row is greater than the last element of the previous row.

Example:

Input:

```
m = 3
n = 4
k = 7

Matrix = [ 1  2   3   4  ]
         [ 5  6   7   8  ]
         [ 9  10  11  12 ]
```

Input format:

```
<m> <n>
<k>
<matrix>
```

Example:

```
3 4
7
1 2 3 4
5 6 7 8
9 10 11 12
```

Output:

```
True
1 2
```

Note: You can use either C++ or Python to code.

SUBSECTION B.2: COMPUTER VISION

OpenCV is the most used image processing library on the planet! So it is a crime to not include any questions from the same topic!

1. We want our vehicle to avoid obstacles and potholes on the road. Write a code to detect the simulated potholes in the videos in the following drive link.

   https://drive.google.com/drive/folders/1ArXXzfO5eJ07TVH_UC9mm-JgaHEkIl3R?usp=sharing

   Your objective is to detect the potholes by drawing bounding boxes over them in the input video itself

   You can use either C++ or python for this.(Python is more user friendly for OpenCV though)

SUBSECTION B.3: Literature Review and Theory

1. What are the different sensors that are typically used to obtain the data which is fed into the navigation stack for autonomous systems? What are the advantages and disadvantages associated with these sensors? When do we use which? What about sensor fusion? (Answer in around 200 words. Be clear and concise. Use diagrams if required.)

2. Below is a research paper which talks about detecting intersections given a bird’s eye view image (obtained through some manipulations from the image from the stereo camera).

   Link to the paper: 3D LIDAR Point Cloud based Intersection Recognition for Autonomous Driving.

- Make a brief documentation on your understanding of the above paper. The documentation can include insights, results, alternatives and any suggestions/improvements that you can think of. We essentially want you to explain the algorithm to us, discuss how well it will/won’t work.

BONUS Questions (OPTIONAL)

Question 1

Implement the algorithm mentioned in the paper in Python and test on the below images (Classifier need not be built, algorithm only for obtaining the ray number vs normalized distance graph will suffice)

Above paper talks with respect to the LiDAR data. The same idea can be implemented on an image as well.

Link to the images: Click Here

Question 2

One of the key factors in establishing a fruitful association with a sponsor is crafting an effective and impactful sponsorship email. Draft an email to a company/organization of your choice, inviting them to sponsor Team Abhiyaan. Choose the company appropriately. Be clear on what you expect from the sponsorship and how you would like the discussions, dealings, and agreements to go ahead. Also, Mention why you feel that the company you have chosen is the right fit to become a sponsor of Team Abhiyaan. Keep in mind that the chosen company/organization should not be an existing sponsor of the team.

Question 3

Make an Instagram poster (.png/.jpg/.jpeg file) welcoming the sponsor of Team Abhiyaan. Visit our team website and social media profiles for any additional information. Here are the links for your reference:

Instagram
Website
YouTube
LinkedIn
Facebook

-------------------x————————————————————————x——————————