CS Quizam (50 pts)

1. (20 pts) Robot Control Architectures
   a. Briefly define the 2 robot control architectures: 1) reactive and 2) deliberative
   b. Which one would have difficulty with the task of delivering mail in an office building – you must explain why to get credit.
   c. Which one works best with the Closed World Assumption – you must explain why to get credit.
   d. What type of robot control architecture is your robot using for the final project? To get full credit you must describe the design of your robot’s control architecture.

A. 1) Reactive
   ![Stimulus → Behavior → Response](image)
   Tightly couples perception to actions
   No intervening abstract representation

2) Deliberative
   ![Sense → Plan → Act](image)
   Senses the world, constructs a world model, creates a plan, makes an action, then senses the results of the action

B. Reactive - it does not have a world model to construct a plan for effectively finding a path to deliver the mail to the correct office.

C. Deliberative - since it plans an optimal solution it could execute it accurately assuming no changes or surprises (closed world assumption)

D. Hybrid - must indicate how reactive and deliberative work together.
2. (10 pts) Repeat the A* search presented in class, but this time start in Timisoara (the goal is to get to Bucharest). Show the search tree. Assume you have written the program to avoid revisiting cities along the same path.

- For each node in the tree show the calculation of the evaluation function, f(n).
- For each level of the search tree show the open list of nodes.
- Diagram is on the last page (you may detach the page).
3. (10 pts) Assume an interleaved image representation in RGB color planes. Write the pseudo code for image segmentation. The input to your function will be a specific area within the RGB color cube.

```plaintext
for (i = 0; i < numberRows; i++)
    for (j = 0; j < numberColumns; j++) {
        if (((ImageIn[i][j][RED] >= redValueLow)
            && (ImageIn[i][j][RED] <= redValueHigh))
            && ((ImageIn[i][j][GREEN] >= greenValueLow)
                && (ImageIn[i][j][GREEN] <= greenValueHigh))
            && ((ImageIn[i][j][BLUE] >= blueValueLow)
                && (ImageIn[i][j][BLUE] <= blueValueHigh))) {
            ImageOut[i][j] = 255;
        } else {
            ImageOut[i][j] = 0;
        }
    }
```