Software Project - Assignment 1
Programming for Engineers 2021/22 - MSc

[50% of total module marks]

Overview

This is an individual assignment worth 50% of your *Programming for Engineers* module marks. The task is to construct a search program in C that implements an intelligent mechanism for a cleaning robot to find the shortest route from its current position to its charging station.

Problem description

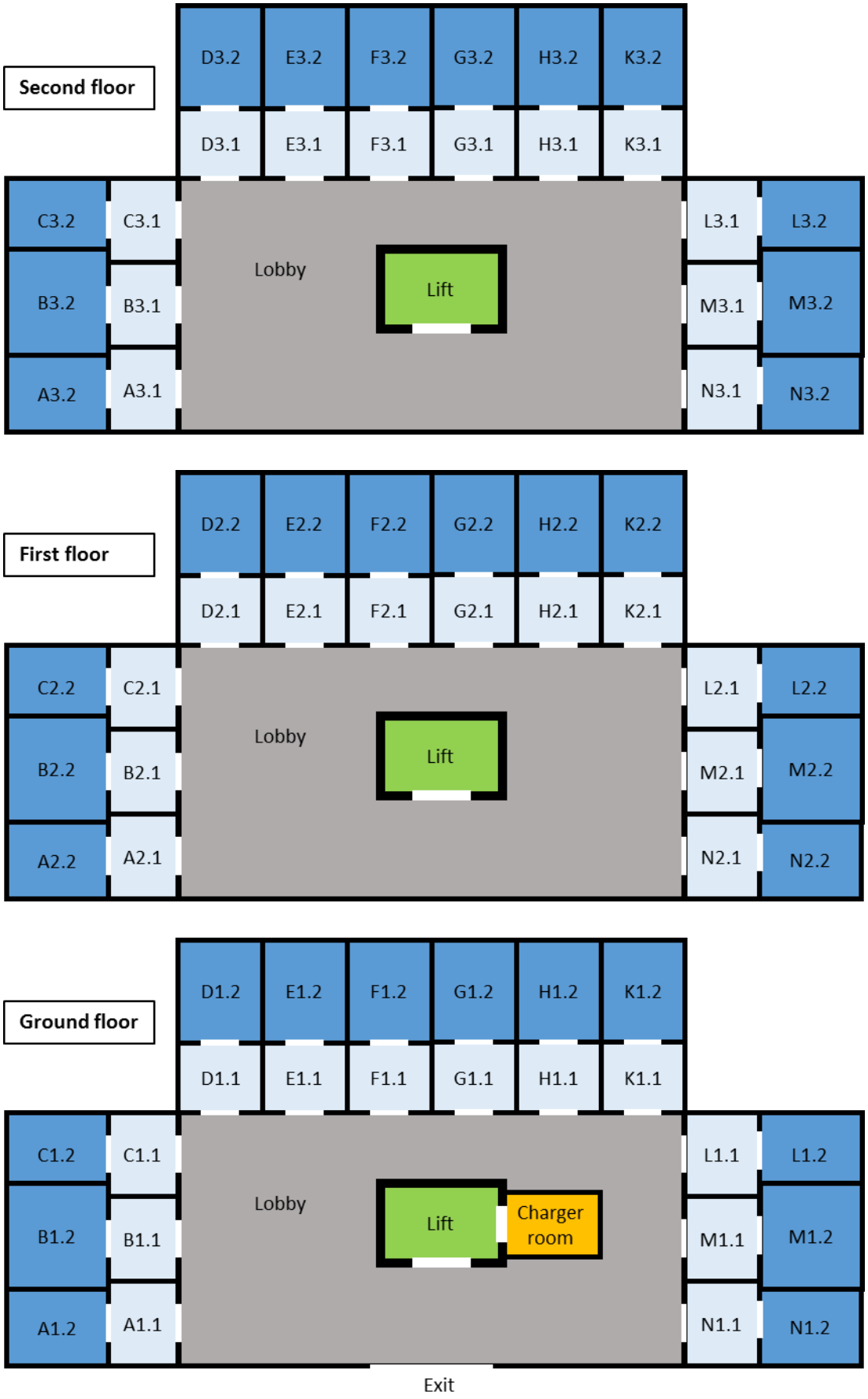
Meet *HOOVI*, a battery-operated automatic vacuum cleaner robot.
HOOVI operates during the night and its task is to keep clean all locations in an office building. The building has three floors with a **lobby** as well as **12** offices and **12** reception rooms on each floor. There is also a **lift** for travelling to the next floor below or above.

Please consult the floor plans on the next page for details on how HOOVI can travel between locations. Looking at these details, note that access to any office is only possible through its dedicated reception room – for instance, on the second floor the office D3.2 can only be reached via the reception room D3.1.
Room labels indicate pairs of connected reception & office rooms, as well as the floor number. When developing your algorithm, take note of the common format in which these labels are written. Each of the reception rooms has access to the lobby on that floor which must be crossed on the way to the lift. (Tip: it may be a good idea to consider the lobby and the lift on each level as separate rooms).
The charging station is in a separate room on the ground floor, accessible directly from the lift. HOOVI can use the lift to travel between floors but only by one floor at each step.

Here more details about how HOOVI operates. You may use any of this to inform your program design, but you also need to decide what is relevant for solving the problem and what is not:

1. HOOVI normally starts exploring the building by going from the charging room into the lift and then travels to all locations (rooms, lobbies, lift) it can find a way into.
2. HOOVI has sensors to assess whether a room is dirty or clean and will only clean up those rooms perceived as dirty. If a room is clean, it will move straight on to the next room.
3. HOOVI monitors its battery level whilst travelling or cleaning. If the battery level becomes critically low, HOOVI should return to the charging station on the ground floor immediately.
4. Even though HOOVI will generally travel along the same route during cleaning, the functionality described in 2. means that in the event the robot needs to return to the charger, it may be in any room on any floor when it needs to return to the charger room.
5. HOOVI can determine the location in which it currently is and has a basic built-in model of the building (that tells it where it can go from a given room), as illustrated in the floor plans.
6. HOOVI must perform a simulated search from the place it is currently in to find the charging room. It has sufficient memory to remember any rooms it has considered going to during that search.

Floor plans


Second floor
D3.2
E3.2
F3.2
G3.2
H3.2
K3.2
D3.1
E3.1
F3.1
G3.1
H3.1
K3.1
C3.2
C3.1
Lobby
Lift
L3.1
L3.2
B3.2
B3.1
M3.1
M3.2
A3.2
A3.1
N3.1
N3.2
First floor
D2.2
E2.2
F2.2
G2.2
H2.2
K2.2
D2.1
E2.1
F2.1
G2.1
H2.1
K2.1
C2.2
C2.1
Lobby
Lift
L2.1
L2.2
B2.2
B2.1
M2.1
M2.2
A2.2
A2.1
N2.1
N2.2
Ground floor
D1.2
E1.2
F1.2
G1.2
H1.2
K1.2
D1.1
E1.1
F1.1
G1.1
H1.1
K1.1
C1.2
C1.1
Lobby
Lift
Charger room
L1.1
L1.2
B1.2
B1.1
M1.1
M1.2
A1.2
A1.1
N1.1
N1.2
Exit

Software Project - Assignment 1
Programming for Engineers 2021/22 - MSc

[50% of total module marks]

Task

Your task is to write a search program for HOOVI's hardware platform with which it can find the shortest path (i.e. least number of rooms travelled) from its current location to the charger room. Extract the relevant problem-solving knowledge from the description above and devise a suitable state representation to capture the information about locations. Devise a non-trivial successor function that determines where HOOVI can go from any given place. For testing your program, you should assume three scenarios and present the respective solution paths in your report, where HOOVI is

I) in room K3.2. **II) in the lift on the second floor.** **III) in room F1.1**

Marking Criteria

There are two groups of assessment criteria, **Program Internals** (search algorithm) and **Human-Computer-Interface** (i.e. input and output on the command line interface). Each of the criteria in these groups is worth a different amount of marks (Total: 100).
When developing your program, you should try to satisfy as many of these as you can. In the event that you couldn't manage to get your program to run, you can still get some marks for any sensible elements that might form part of a correct solution. Note that satisfying a criterion somehow does not automatically yield the stated marks - there may be deductions for inefficiencies and mistakes made, or overly cumbersome approaches. While you are not primarily marked on the quality of your code, it is advised to adhere to common practices. This includes sticking to the specification, using suitable data types for representing the information and for storing results, avoiding undue code duplication, and most importantly: use of comments in the code. Some features require you to think about user interaction – make sure your program presents information clearly and is easy to use.

Program Internals (70 marks in total)

1. Appropriate state representation. **(8)**
2. Correct initial state defined. **(2)**
3. Core elements of a correct successor function that generates subsequent states. **(4)**
4. A successor function that generates valid states for solving the problem. **(10)**
5. A successor function that generates valid states such that HOOVI can use the search sequence as a cleaning pattern for its routine job (all rooms to be searched). **(16)**
6. Correct goal test. **(2)**
7. Some kind of search activity going on. **(5)**
8. Correct elements of Breadth-First-Search list/queue management. **(6)**
9. Overall correct and efficient BFS implementation. **(10)**
10. An alternative working Depth-First-Search variant accessible within the same program **(7)**

Human-Computer-Interface (30 marks in total)

1. A text output of the starting location. **(2)**
2. A text output signaling that a solution path has been found. **(2)**
3. A text output of the goal location. **(2)**

4. Detailed printing of the solution path along the different locations involved. **(6)**
5. Detailed status messages about HOOVI's search activity that can be switched on and off through a variable in the code or by user input at the beginning of the program. **(11)**
6. The user can enter an arbitrary initial location at the start which is then used by the algorithm to determine a solution path from there. **(5)**
7. The user can choose between BFS and DFS as search strategies from a menu. **(2)**

Report

Your report is a very important element of the assessment to confirm that you have written the program yourself and understood the functionality implemented in it. It will hence be used to qualify the marks you receive. Your report should point out the relevant marking criteria that you have addressed, with detailed references to your source code, and answer the questions posed. If you document your code extensively during development, then this will be a lot easier to do and you need less explanation in the actual report text. For this reason, there is no minimum word count given, but as a guideline, try not to exceed 1500 words in total (excluding Appendix).

Your report should have a title page with your name and candidate number on it and contain the following chapters:

1. **Introduction** – A short section (no more than half a page) where you should describe the purpose of the program and list the main functions in the code, briefly pointing out what they do. You may then also include a screenshot of the program window, if you like.
2. **Program description** – here, you need to explain how your implementation addresses the marking criteria. Focus on the *Program Internals* and make references to the code in the appendix - you do not need to describe any of the interface criteria as these are self-evident. Which search strategy do you think is most suited to this task – BFS or DFS? Explain why.
3. **Appendix** (source code as text, ideally with line numbers for easier reference from chapter 2)

Submission Requirements

Your submission must contain the following parts:

- **Your program (source code file(s))** – must be zipped up into a single ZIP file!
- **Your report as PDF** (included in the ZIP file)

You should submit your coursework online via Canvas. Keep in mind that this is an individual assessment - the related University rules on academic misconduct apply, and here in particular with respect to collusion with fellow students and plagiarising code from Web pages or elsewhere.

Submission Deadline

The assignment is due on Friday, 26 November 2021, by 4pm.

Good luck!