TAU - Advanced Topics in Programming, Semester B 2024

Assignment 1

Note:
This document is in DRAFT mode till June-13th. While in draft mode, it may have significant changes without notice and without highlighting them. Even though it is still a draft, you may start working on the exercise.
After the draft period there could still be changes but then they would be highlighted in the document.
During the draft period, you are mostly welcomed to add questions directly into this document, as comments. Please assign your comments and questions to kirshamir@gmail.com so I will get an email when you add them. Then please follow my answer and if it resolves your comment or question please close it.
After the draft period, commenting on the document will be closed and all questions on the assignment would be submitted in the relevant forum in the course site!

Submission deadline: June-30th, 23:50.
Note: here are the Submission Instructions - please make sure to follow them!

Assignment 1 - Requirements and Guidelines

You are required to implement a program that simulates an automatic vacuum cleaner.

In this first assignment the requirements are a bit open, you will have to select your API, class design and input file format. Note that in the next assignments a common API and file formats will be dictated and you will have to refactor your code accordingly.

House Input File - TEXT file

The program should get an input file that represents a house, with walls and corridors. The house should be surrounded by walls (no need to check that, however, if the house in the file is not surrounded by walls, the program should surround it by walls).
The house has 10 levels of dirt per each point in the house, from 0 - no dirt to 9 - very dirty. You can decide to mark that in the file with the digits 1 to 9 for dirt and space for no dirt.
The house input file should also contain the location of the docking station from which the vacuum cleaner sets off and gets back for charging.
In addition, two configurations should be also in the house input file:

- Number of “steps” till the battery is off (“max battery steps”).
- Maximum number of “steps” to allow for completing the mission before stopping (if this maximum is reached we stop, even if the mission is not completed).

Steps, Charging, Cleaning

A step is counted as a move of the vacuum cleaner, or staying in place (e.g. for cleaning dirt at that location or charging). When the vacuum cleaner stays in place, if this position has dirt, the

dirt level is decreased by one. The vacuum cleaner finishes its mission when there is no dirt in the entire house and it is back at the docking station. Note that this result can be achieved without visiting the entire house and it is considered a legitimate finish, even if achieved "accidentally".

When the program starts the vacuum cleaner is at its docking station and fully charged.
If the vacuum cleaner reaches docking station, its new charge is calculated as:

(the amount of battery steps when getting onto the docking station) +
(number of steps staying on the docking station) * ("max battery steps" / 20)

According to the above formula, staying for 20 steps at the docking station should fully charge the battery, even if it was completely empty when getting onto the docking.

If the charging is exhausted and the vacuum cleaner is not in the docking station then the program should end with a failure for the vacuum cleaner, even if the house is fully cleaned (the way to communicate the results is elaborated later).

Algorithm

The algorithm of moving around and cleaning the house should be isolated from all the rest of the program. It should certainly NOT be aware of the house structure. The rest of the program should also not use any information from the algorithm, except for the movement decision the algorithm will have per each step.
In order to decide on the next step, the algorithm can query the following sensors: (1) a dirt sensor that reports the level of dirt in the current location. (2) a wall sensor that reports whether there is a direct wall in each of the directions North, East, South, West. (3) a battery sensor, returning the amount of steps left (you may assume the battery sensor is always accurate). Your design should think of the API that will tie between the algorithm and the sensors.

A good algorithm should probably build its own map of the house, while exploring it, and try to efficiently cover the entire house (using an internal algorithm such as BFS or DFS).
HOWEVER, in your first assignment you are not required to implement any smart algorithm.
We expect you to implement a naive algorithm that:

- will not move into walls
- will not stay in a location with no dirt (except for docking)
- will not stay in docking more than required
- will refer to the battery state (will remember the way back to the docking station and get back before the battery is exhausted)

The moves in the house can be random in your first assignment, except for when getting back to the docking station.

The algorithm should be able to communicate its next requested step for the vacuum cleaner (one of: North, East, South, West, Stay). It can then assume that the requested step was performed. Note that the algorithm should not write anything into the output file, it should communicate back its move decisions based on your API.

Output File - TEXT file

The output file should include:

- all the steps performed by the vacuum cleaner
- the total number of steps performed (yes, we can count it from the previous item, but we still prefer to have it as a separate additional output)
- a result of the amount of dirt left in house
- an indication of whether the vacuum cleaner is dead (battery exhausted)
- an indication of whether mission is succeeded (no dirt, vacuum cleaner at docking)

Error Handling

You should reasonably handle any error:

- Program shall not crash, never.
- You should better recover from errors that are recoverable, and continue based on reasonable decision making.
- In case there is an unrecoverable error, a message should be printed to screen before the program finishes.
- You should not end the program using the “exit” function or any other similar function that ends the program abruptly. The program should always end by finishing main. Even in case of errors.

Running the Program

myrobot <house_input_file>

Additional required documents:

Add a short high level design document (HLD) as PDF, containing:

- A UML class diagram of your design.
- A UML sequence diagram of the main flow of your program.
- Explanations of your design considerations and alternatives.
- Explanations of your testing approach.

Bonuses

You may be entitled for bonus points for interesting implementation.
NOTE that implementing a smart algorithm will not be entitled for a bonus, as this would be part of your next assignments.
But, the following may be entitled for a bonus:

- adding logging, configuration file or other additions that are not in the requirements.
- having automatic testing, based on GTest for example.
- adding visual simulation (do not make it mandatory to run the program with the visual simulation, you may base the visual simulation on the input and output files as an external utility, it can be written in C++ or in another language).

IMPORTANT NOTE: In order to get a bonus for any addition, you MUST add to your submission, inside the main directory, a *bonus.txt* file that will include a listed description of the additions for which you request for a bonus and how to check them.