ESE 124 Programming Fundamentals
Prof. Alex Doboli

Course Project
Spring 2024

Michael: The Fantastic Ant

Implement a C program with the following description.

1. The description of VA (Virtual Ant) – a fantastic ant

Michael, our compassionate ant, is a virtual ant (VA). For brevity, the VA is called from now on as just Michael.

- Michael has its own memory implemented as a stack of size *MAX_SIZE* and with the operators (functions) like those discussed in class: pop, push, peek, clear, empty, and full. Each stack element can store the current physical position of the ant inside the maze. The physical position is described as the pair **x,y**. The positions are indicated by 2 integer numbers, **x** and **y**, but the accurate position is not displayed to Michael. So, Michael does not know the map coordinates of the place it must navigate and it does not know its current position coordinates or any of its previous positions' coordinates.
- Michael knows to perform the following actions [Note that **x** is the row number and **y** is the column number in the following description]:
    1. MARK – the ant marks its current position using a chemical called pheromone.
    2. MOVE_F – moves the VA from the current position one position forward. If Michael locates at coordinates (x, y), it will move to (x + 1, y).
    3. MOVE_B – moves the virtual ant from the current position one position backward. If Michael locates at coordinates (x, y), it will move to (x - 1, y).
    4. MOVE_L – moves the virtual ant from the current position one position left. If Michael locates at coordinates (x, y), it will move to (x, y - 1).
    5. MOVE_R – moves the virtual ant from the current position one position right. If Michael locates at coordinates (x, y), it will move to (x, y + 1).
    6. CWL – Michael checks if the next locations to the left (until meeting a wall) are pheromone free. If the locations are free then Michael feels an itch. Otherwise, if no location is free (e.g., because there is a pheromone mark or a wall on the left of Michael), then Michael does not feel the itch.
    7. CWR – Michael checks if the next locations to the right (until meeting a wall) are pheromone free. If the locations are free then Michael feels another kind of itch.

Otherwise, if no location is free (e.g., because there is a wall or a pheromone mark on the right of Michael), then Michael does not feel the itch.

8. CWF – Michael checks if the next locations in front (until meeting a wall) are pheromone free. If the locations are free then Michael feels a third kind of itch. Otherwise, if no location is free (e.g., because there is a wall or a pheromone mark in front of Michael), then Michael does not feel the itch.

9. CWB – Michael checks if the next locations backwards (until meeting a wall) are pheromone free. If the locations are free then Michael feels a fourth kind of itch. Otherwise, if no location is free (e.g., because there is a wall or a pheromone mark behind of Michael), then Michael does not feel the itch.

10. PUSH – pushes the planar coordinates **x** and **y** of Michael's current position into Michael's stack for the memory. This way Michael memorizes the current position.

11. POP - pops the planar coordinates **x** and **y** from the top of the Michael's stack for the memory. This way Michael remembers the position, but then it immediately forgets the position too.

12. PEEK - peeks the planar coordinates **x** and **y** from the top of the Michael's stack for the memory. This way Michael remembers the position but does not forget the position.

13. CLEAR – Michael clears its stack. This way it has a total amnesia and forgets all its previous memories.

14. BJPI (Bold jump for itching) – jump **x** position along the direction for which Michael felt an itch (left, right, forward, backward). For example, after performing CWR, Michael felt an itch because the direction to the right of the current position was free. Then it decided to act using BJPI, meaning that it jumped **x** positions to the right, as the itching was felt after Michael checked the locations to the right of its current position. The number of positions over which it jumped is found by after using the corresponding CWL, CWR, CWF, CWB action. For example, if the next 4 locations to the left of the current position are free, then **x** is 4 after executing CWL. If **x** is zero after executing action CWL, but the ant still executes action BJPI, then Michael stays in its current position. Every BJPI stops the corresponding itching of the ant, e.g., the itching type that triggered the jump.

15. CJPI (cautious jump for itching) – jump **one** position along the direction for which Michael felt an itch (left, right, forward, backward). For example, after performing CWR, Michael felt an itch because the direction to the right of the current position was free. Then it decided to act using CJPI, meaning that it jumped one positions to the right, as the itching was felt after Michael checked the locations to the right of its current position. Every CJPI stops the corresponding itching of the ant.

16. BACKTRACK – Michael backtracks to the position that is retrieved from the memory, such as using POP or PEEK. Example: Michaele executes a pop and finds the position <4,4>. Backtrack will move Michael back to this position.

17. RP n t – repeats the **n** actions following the RP action for **t** times.

2. Functionality description

- Good deeds are placed at different positions inside a maze. For example, a good deed is a bag of food that can be donated to hungry people, or a certain activity that can be performed to help homeless persons. Each good deed values a certain number of points, some more and other less.
- Michael must traverse a labyrinth (maze) and find as many good deeds as possible within a given number of steps. Therefore, it must use an intelligent strategy.
- Michael's intelligence is described as a sequence of actions that Michael executes to find its path to the good deeds. You (meaning you, the student taking ESE 124) describe the sequence of actions in an input file, which your C program reads before requiring Michael to execute the actions. So, arguably, the input file is Michael's brain which holds its intelligence defined as the actions that VA executes to find good deeds while traversing the maze.
- Michael does not know beforehand which of the good deed carries more points and which less. But there is a hidden rule according to which the deed points are assigned based on the position of the deeds, e.g., their **x** and **y** coordinates. Can Michael learn the hidden rule while navigating the maze, so that it maximizes the number of points of the collected good deeds?
- The labyrinth has one entry and Michael enters the labyrinth using the entry.
- The maximum number of steps that VA can execute it the constant *MAX_NUMBER_OF_STEPS.*
- The labyrinth to be traversed is read from a file.
- The sequence of actions performed by Michael is written into an output file.

3. Other program requirements

- Your program must use an Abstract Data Type (ADT) VA that implements all the actions of the ant. Your program must also create an Abstract Data Type (ADT) for the ant's memory.

4. Experiments

- Execute the program for different values of constants *MAX_SIZE* and *MAX_NUMBER_OF_STEPS*.
- Consider different values for *n* and *t* in the actions RP n t. Record for each case, the number of found good deeds, their total number of points, and the number of performed steps.