Project 1 Design + Reflection
Tay, Serena
Student #: 933588509

Document (Design + Reflection)

You need to create your program design **BEFORE** you start coding. Your design needs to be included in your document.

Simply stated, **program design** is identifying the problem to be solved, modularizing and describing the structure of program, and making a test table to that fulfills the requirements of the program.

For test table, please include test plan, expected output, and actual output. The test plan should be robust. The expected output is based on the requirement. If your actual output deviates from your expected output, that usually means the program behavior is deviating from the requirements. Following is a Examples Test Plan.pdf

In your **reflection**, you should explain what design changes you made, how you solved some problems you encountered, and what you learned for this project. The TA will grade on how well your implementation matches your design.

Design

Create three classes with the following functions

- Simulation class – As the 'master class' that dictates the flow of the program
- Menu class – To display initial menu and a prompt function that prompt users for inputs for the program
- Ant class – stores all the users' input, the board data and the Langton's ant rule. This class will then use these inputs for the ant program

Ant Class

Contain the following member variables to store user inputs and board data:

- rowCount
- columnCount
- stepCount
- rowLocation
- columnLocation
- orientation
- colour (black or white)
- char** board (2D dynamically allocated array to create and print board)

Contain the following class functions:

- Ant() – a default construct to initialize all member variables

Project 1 Design + Reflection
Tay, Serena
Student #: 933588509

- Ant(int rowCnt, int columnCnt, int stepCnt, int startingRowCnt, int startingColumnCnt) – a constructor that initialize an Ant object with user inputs gathered by simulation class (to be used in simulation class)

- void setOrientation (char) – to be used by move ant to set the orientation for each move
- void printOrientation(int) – to print orientation should needed for testing purposes

- void setColour (char) – to be used by move ant to set the colour of the space for each move
- void saveNextStepColour () – used to store the next step before placing ant on it

- void placeAnt () – place ant when initiating game
- void stepForward() – move ant one step forward, used in move ant function
- void moveAnt () – function that contains all the functions within the ant class to be used to run the Langton's ant rule and in sequence

- void createBoard () – to create 2D dynamically allocated board based on input provided by user
- void printBoard () – print board after each move
- void deleteBoard () – delete board after program ends to prevent memory leak

Data validation/integrity consideration for Ant class:
- make sure Ant does not go out of bounds from the board to prevent memory leak
- utilize the board wrap provided on Canvas (cite it)
- make sure functions are limited by the size of the board (col and row count per user input)

Menu Class
Contain the following member variables:
- countPlay – This is used to determine if the "new run" menu should be used or "play again" menu should be used per requirements
- menuChoice – used to store choice entered by user

Contain the following member functions:
- Menu () – default constructor to initialize member variables used
- displayMenu() – to display menu (contained 2 menus, one for the new game, one for the 'play again' scenario once the first simulation run occurred
- prompt(string, int, int) – a generic function that can be used to prompt user for data inputs. Contain 3 parameters, one string to tell user which member variable the program is asking for, one integer for the minimum allowable input and one for the maximum allowable input

Project 1 Design + Reflection
Tay, Serena
Student #: 933588509

- getMenuChoice() – function to be used in simulation class to obtain user's input

Data validation/integrity consideration for Menu class:

- make sure user gets an error message when entering an invalid choice
- create a max and min for a range of allowable input (create an error message when user entered an invalid number outside the range)

Simulation class
Contain the following member variables:

- choice – get choice from menu class to be used to determine what the user wants (to play or to quit)

The following member variables are all used to store user inputs to be passed into the Ant class using the Ant class constructor

- rowCount
- columnCount,
- stepCount,
- startingRow,
- startingColumn;

Contain the following member functions:

- Simulation() – default constructor used to initialize all the member variables listed above
- runSimulation() – used to run the simulation and calling the menu and Ant class in sequential order (display menu, ask user for input, run program, ask user what they want and repeat until user wants to quit)

The following member functions are used by the Ant class constructor to pull user inputs from the simulation class to initialize member variables in the Ant class:
getRowCount()
getColumnCount()
getStepCount()
getStartingRow()
getStartingColumn()

Data validation/integrity consideration for Simulation class:

- Use menu functions to ensure user inputs are valid

Project 1 Design + Reflection
Tay, Serena
Student #: 933588509

Project 1 Design + Reflection
Tay, Serena
Student #: 933588509

Testing plan for each function:
Ant Class

- Ant() – a default construct to initialize all member variables

Test Case	Input Values	Expected Outcomes	Observed Outcome
Ensure constructor initialize to default values	N/A – set in program	Member variables initialize to the values stated	Member variables initialized to the values stated

- Ant(int rowCnt, int columnCnt, int stepCnt, int startingRowCnt, int startingColumnCnt) – a constructor that initialize an Ant object with user inputs gathered by simulation class (to be used in simulation class)

Test Case	Input Values	Expected Outcomes	Observed Outcome
Ensure constructor initialize to values passed from Simulation class	getRowCount() getColumnCount() getStepCount() getStartingRow() getStartingColumn()	Member variables initialise to the values stored in Simulation class	Member variables initialised to the values stored in Simulation class

- void setOrientation (char) – use printOrientation to test if this function is producing the right result

Test Case/Input Value	Expected Outcome	Observe Outcome
L – turn left	1. Cause orientation to turn left 2. Cause orientation to turn to left when at up	1. Success – Achieved expected outcome 2. Success – Achieved expected outcome
R – turn right	1. Cause orientation to turn right 2. Cause orientation to turn up when at left	1. Success – Achieved expected outcome 2. Success – Achieved expected outcome

- void printOrientation(int) – to print orientation should needed for testing purposes

See test above

Project 1 Design + Reflection
Tay, Serena
Student #: 933588509

- void setColour (char) – to be used by move ant to set the colour of the space for each move

Test Case/Input Value	Expected Outcome	Observe Outcome
W – turn space to white	Switch current space to white	Success – Achieved expected outcome
B – turn space to black	Switch current space to black	Success – Achieved expected outcome

- void saveNextStepColour () – used to store the next step before placing ant on it

Test Case/Input Value	Expected Outcome	Observe Outcome
Save next step colour before placing ant on it	1. Save black when the next step is black 2. Save white when the next step is white	Success – Achieved expected outcome Success – Achieved expected outcome

- void placeAnt () – place ant when initiating game

Test Case/Input Value	Expected Outcome	Observe Outcome
Ensure ant is placed on the correct location based on users input of starting row and column	1. row placed = startingRow 2. column placed = startingColumn	Success – Achieved expected outcome Success – Achieved expected outcome

- void stepForward() – move ant one step forward, used in move ant function

Test Case	Input Values	Expected Outcomes	Observed Outcome
Move one step forward in the direction of the orientation	Orientation: 1 2 3 4	Move one step in the orientation direction for all directions	Success – Achieved expected outcome for all directions
When the ant is at the end of the board, make sure it wraps around	Which end: Up Right Left	When Orientation of ant for next step goes to the border of the board Up – Jump to bottom Right – Jump to left	Success – Achieved expected outcome for all directions

	End	Down – Jump to top Left – Jump to right	

- void moveAnt () – function that contains all the functions within the ant class to be used to run the Langton's ant rule and in sequence

Test Case	Input Values	Expected Outcomes	Observed Outcome
Ant on white space move according to rule	From member variables in ant class	1. Turn right 90 degrees 2. Set current space to B 3. Step forward one space based on current orientation 4. Save the next step's color 5. Place ant on new location 6. Prints board with updated results	Success – Achieved expected outcomes
Ant of black space move according to rule	From member variables in ant class	1. Turn left 90 degrees 2. Set current space to W 3. Step forward one space based on current orientation 4. Save the next step's color 5. Place ant on new location 6. Prints board with updated results	Success – Achieved expected outcomes

- void createBoard () – to create 2D dynamically allocated board based on input provided by user

Test Case	Input Values	Expected Outcomes	Observed Outcome
Make sure function creates a board with the minimum allowable range (5)	Row count Column count	Rows and Column matches member variables and no leaks	Success – Achieved expected outcomes

Project 1 Design + Reflection
Tay, Serena
Student #: 933588509

Make sure function creates a board with the maximum allowable range (100)	Row count Column count	Rows and Column matches member variables and no leaks	Success – Achieved expected outcomes

- void printBoard () – print board after each move

Test Case	Input Values	Expected Outcomes	Observed Outcome
Make sure function prints board based on the initial user inputs	Row location Column location Orientation Colour of space	1. Rows and Column matches member variables 2. No memory leaks 3. Ant placed on correct location	Success – Achieved expected outcomes
Make sure function prints board based on the updated inputs after the moveant function runs through the Langton Ant rule every step	Row location Column location Orientation Colour of next space	1. Rows and Column matches member variables 2. No memory leaks 3. Ant placed on correct location	Success – Achieved expected outcomes

- void deleteBoard () – delete board after program ends to prevent memory leak

Test Case	Input Values	Expected Outcomes	Observed Outcome
Ensure board is deleted	N/A	No memory leaks	Success – Achieved expected outcomes

Menu class

- displayMenu() – to display menu (contained 2 menus, one for the new game, one for the ‘play again’ scenario once the first simulation run occurred

Project 1 Design + Reflection
Tay, Serena
Student #: 933588509

Test Case	Input Values	Expected Outcomes	Observed Outcome
Display 1st menu when program initialize	Menu counter	Display first menu	Success – Achieved expected outcomes
Display "play again menu" after first run	Menu counter	Display play again menu	Success – Achieved expected outcomes

- prompt(string, int, int) – a generic function that can be used to prompt user for data inputs. Contain 3 parameters, one string to tell user which member variable the program is asking for, one integer for the minimum allowable input and one for the maximum allowable input

Test Case	Input Values	Expected Outcomes	Observed Outcome
Display correct output for all prompts used in Simulation class	Simulation class calls	Display appropriate prompts for: rowCount columnCount stepCount startingRow startingColumn	Success – Achieved expected outcomes for all inputs prompted for
Display error messages if input does not fall between the min and max set	Simulation class calls	Display appropriate error message when user input is outside of valid range set and reprompt user until valid input is provided: rowCount columnCount stepCount startingRow startingColumn	Success – Achieved expected error message outcomes for all invalid inputs and reprompt user for valid input

- getMenuChoice() – function to be used in simulation class to obtain user's input

Test Case	Input Values	Expected Outcomes	Observed Outcome

Project 1 Design + Reflection
Tay, Serena
Student #: 933588509

Ensure menu choice retrieved matches user input	Menu choice	Menu choice return matches data stored in member variable	Menu choice returned matches data stored in member variable

Simulation class

- Simulation() – default constructor used to initialize all the member variables listed above

Test Case	Input Values	Expected Outcomes	Observed Outcome
Ensure constructor initialize to default values	N/A – set in program	Member variables initialize to the values stated	Member variables initialized to the values stated

- runSimulation() – used to run the simulation and calling the menu and Ant class in sequential order (display menu, ask user for input, run program, ask user what they want and repeat until user wants to quit)

Test Case	Input Values	Expected Outcomes	Observed Outcome
Ensure function is run the function calls in sequence and keep looping until user wants to quit	N/A – set in program	1. display menu 2. ask user for input 3. run program 4. ask user what they want 5. Repeat loop until user wants to quit	Success – Achieved expected error message outcomes for all invalid inputs and re-prompt user for valid input

The following member functions are used by the Ant class constructor to pull user inputs from the simulation class to initialize member variables in the Ant class. Used the following test case for all the functions:
getRowCount()
getColumnCount()
getStepCount()
getStartingRow()
getStartingColumn()

Project 1 Design + Reflection
Tay, Serena
Student #: 933588509

Test Case	Input Values	Expected Outcomes	Observed Outcome
Ensure functions are returning the right values when called in the ant class	User input	Call returns the user input stored in the member variables	Success - Call returned the user input stored in the member variables