2020

Design Document: Zigzag
By: Harsh Bali

HB

HB

Introduction

For this assignment, I created a **Software Requirement Specification** for Zigzag task, which includes **Gathering Software Requirements (Functional, Non-functional and Addition requirements) and process of Understanding the Finch**. It also includes **Design document, for Zigzag Task,** consisting of **Flowcharts, Pseudocode and User-Interface** prototype.

Software Requirement Specification

Requirement gathering is an essential part of designing, as requirements acts as a backbone for the design. Software requirements can be differentiated into two categories**: Functional Requirements- what the system must do(behaviour) and Non-Functional requirement- how the system should do it (qualities).**

Gathering software requirements

By Tradition, the process of gathering software requirements involves, use of **Stakeholder Analysis** to determine, how the stakeholders' requirements must be interpreted. This helps to **develop use-cases**; thus, **functional requirements are gathered using Use-cases** and then, **the non-functional requirements are discussed & decided by taking functional requirements into an account.**

However, when I first read the task brief, I realised that all the functional requirements can be sourced from the task-brief itself and I don't have to interview the project stakeholders to gather functional requirements.

Functional and Non-Functional Requirements

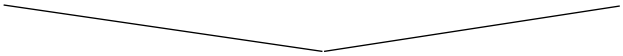
The table below shows the **Functional** and **Non- Functional Requirements** for the **Task4: Zig Zag**. The table also shows **Checklist** columns, that will help me to ensure, that both Functional Requirements and Non-functional requirements have been incorporated in the design.

Functional Requirements	Checklist
1. This program should **only start when the finch is placed on Level floor.**	√
2. The program should make the finch move in **regular zigzag manner**.	√
3. The program should make finch **alternate lights between Green and blue while traversing.**	√
4. The program must ensure **all the zig-zag sections are of same length and are orthogonal to each other**.	√
5. The program should ensure **that user enters length >=15 and length <=85**.	√
6. The program should ensure **that user enters sections >=2 and section <=12**. It should also ensure that **section input should be an even number.**	√
7. The program should **generate error messages when invalid input has been added.**	√
8. The program **must generate speed randomly.**	√
9. The program must be able to **calculate time.**	√
10. The program should **make finch move using calculated time and random speed.**	√
11. The program should make **finch pause for 1 second after finishing each section.**	√
12. This program should make the **finch turn around and retrace the zigzag.**	√
13. The program should make finch use **same pattern of Lights while retracing back.**	√
14. The program should be able to **create a "Report" when finch has stopped retracing.**	√
15. **The "Report" must contain all User Input, random speed, length of traversed path(Not including retrace), total time, total straight-line length.**	√

Non-functional requirements	Checklist
1. The **user interface of the program must be easy to use**.	✓
2. The **error messages generated should guide the user to input correct values.**	✓

Additional Functionalities

In the task, its stated that the **user can only tell finch to make *Even* number of zigzags** (regular zigzag). **For example*: zig zag with 2 sections*** *showed below or* ***the one with 8 sections*** *showed in the task brief.*



However, **zigzags can be irregular too**, like the ***zigzag below with three sections***.

Therefore, I've decided to add **two modes** which are: **Regular and Freestyle mode**. For Regular mode user, the user will be to create zigzag with **only even sections**. Whereas, for the freestyle mode, the user will be able to create zig zag with **even or odd number of zig zag sections**. Moreover, all the **other functional and non-functional requirements for freestyle mode will be the same** and it will also be able to retrace back.

Understanding Finch

In-order to solve the problem, **understanding the problem is mandatory**. As zig zag problem mainly involves using the finch, **I reflected on my understanding of the finch and came to conclusion, that I lack the knowledge on how the finch works**. I also knew, without understanding of the finch, I wouldn't be able to create a robust design for the zigzag problem.

I wanted to understand:

1. **How finch moves for a given distance, using time?**
2. **How finch turns at right angle?**
3. **How the alternating LED lights will work?**

Note: I developed these questions based on core functional requirements directly related to finch.

How finch moves for a given distance?

I found out that finch can't move for given distance, however, it can move for specific time at specific speed. Therefore, I can make use of an equation that can **help me to find time the finch needs to move for a given distance**.

$$Time = Distance \div Speed$$

How finch turns at right angle?

I conducted an experiment to find out at what speed and time the finch turns orthogonal (90 degrees). **I found out finch turns orthogonal when right speed =80, left speed=-80 and time =1000ms.**

Left wheel speed	Right wheel speed	Time(ms)	Observations
100	100	1000	Finch moves forward
-100	-100	1000	Finch moves backwards
100	-100	1000	Finch turns more than 90
90	-90	1000	Finch turns more than 90
80	-80	1000	Finch turns at 90
70	-70	1000	Finch turns less than 90
160	0	1000	Finch turns at 90 but not at same spot.
255	0	500	Finch turns at 90 but not at same spot

How the alternating LED lights will work?

This table helped me to understand how the LED lights should work in this task.

I discovered a pattern, that when the section number was a multiple of two then the lights were blue (even sections are blue).

Zig zag Section NO	Colour
1	Green
2	Blue
3	Green
4	Blue

Another problem encountered

While I was trying to find out the answers for the three questions, that helped me to understand both: the finch and the zig-zag task, **I realised that UNITS for Speed, for finch are missing. Without knowing the units for speed, I wouldn't be able to assign random generated speed to finch. Hence, failing to achieve one of the core functionalities, which is: assigning random speed and time to finch to make it move for specific distance.**

Therefore, I decided to carry out an experiment to find units for speed. The experiment involved use of measuring tape, coloured tape and finch. I stuck the tape onto the floor and observed how far the finch travelled (from tape) at different time while keeping the speed same.

Using this I found out that the units for speed, for finch is **mm/s.**

Flowcharts

Main Flowchart

This is the main flowchart, that shows, my interpretation of the Zigzag task. It also shows, how the various subprocesses are linked with one another to solve zigzag problem.

This flowchart allows to see how the additional requirement can be added without compromising any of the main requirements.

Note: *some of these subprocesses have subprocess within them.*

For example: subprocess ***Zigzag Movement*** consists of ***Finch Movement, Finch Orthogonal Clockwise*** and ***Finch Orthogonal Anticlockwise*** subprocesses. This is because all these subprocesses collectively make finch move in zigzag form.

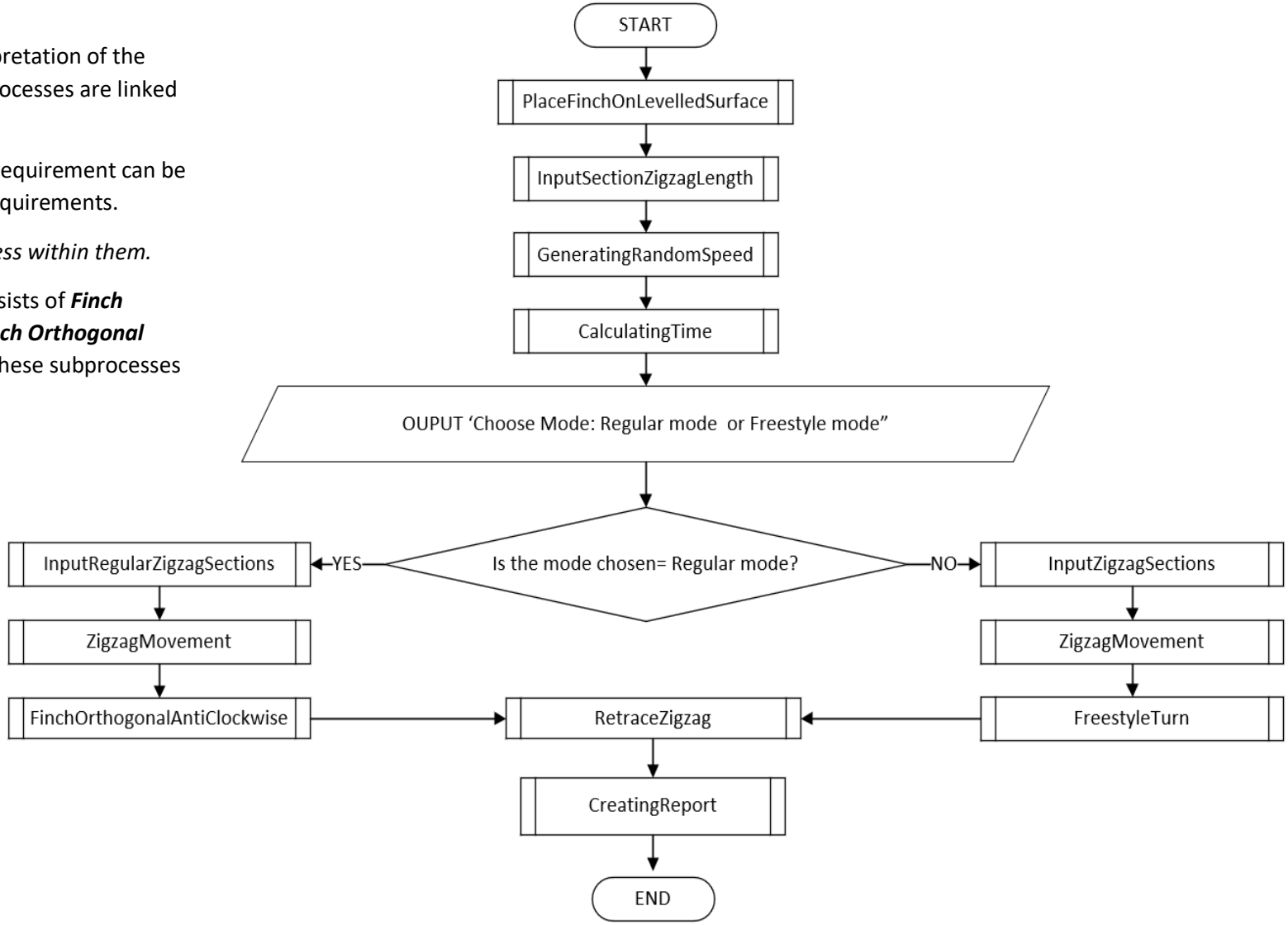
IMPORTANT NOTE:

Both in flowcharts and pseudo code; I've used Pascal Case naming convention for subprocesses and camel Case for variables.

Example:

InputZigzagLength = Pascal Case

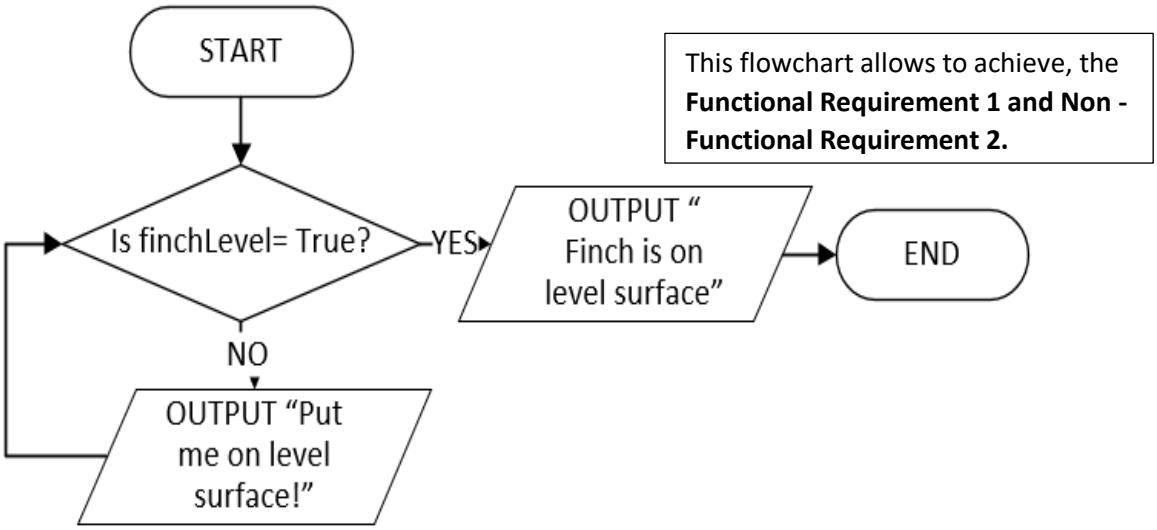
zigzagSectionLength= camel Case



HB

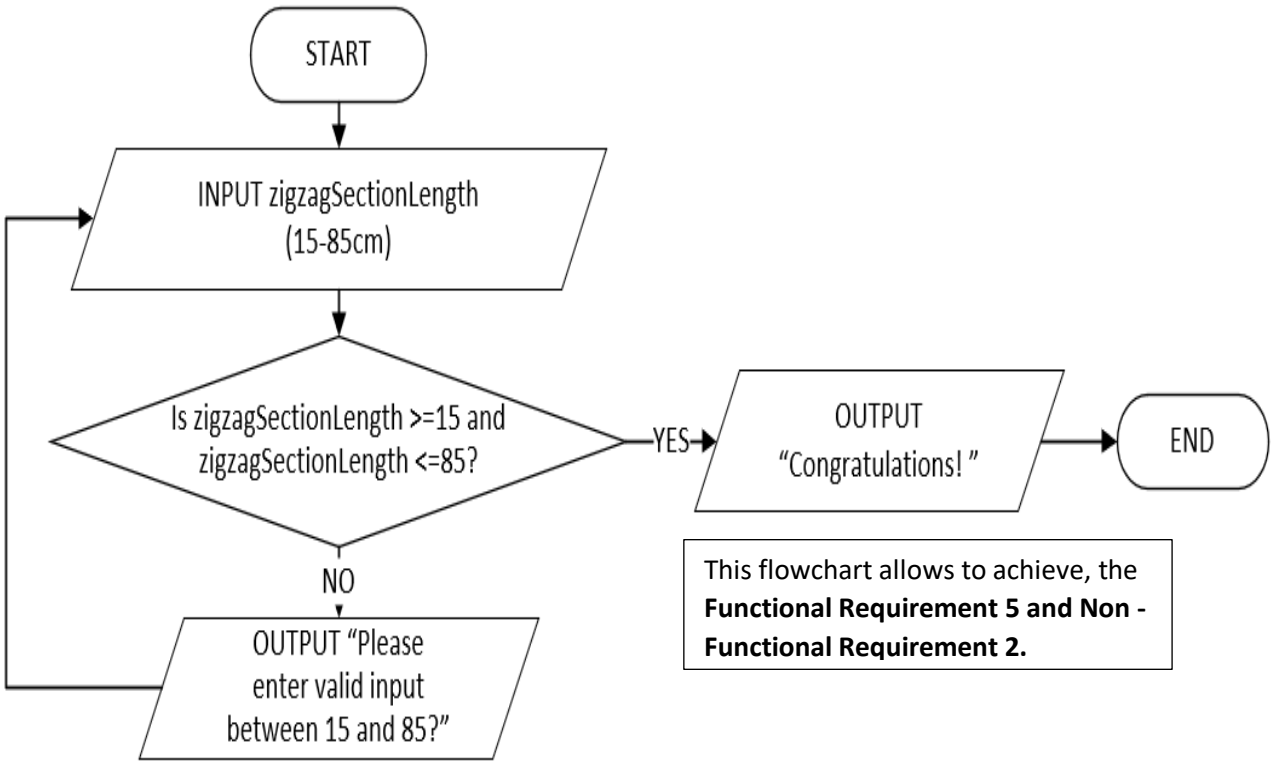
Place Finch On Levelled Surface

In the main flowchart, this is the first subprocess. This flowchart checks with the aid of Finch whether the finch is on levelled surface or not.



Input Zigzag Length

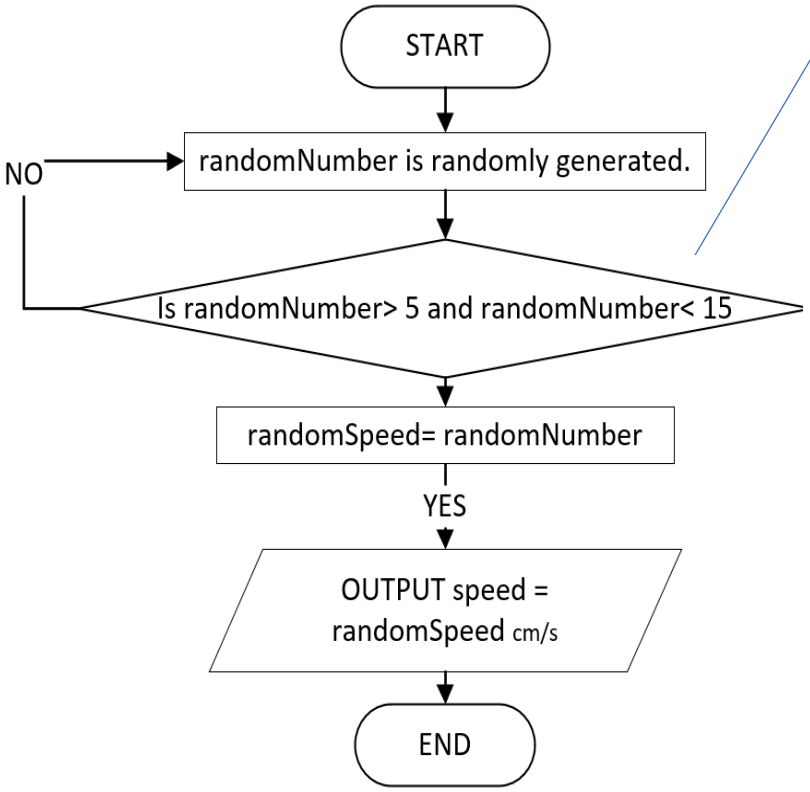
The subprocess below, gets the input from the user (Zigzag Length), if the user enters the valid input (positive integer 15-85), the flowcharts ends with the, "CONGRATULATION!" message. If user enters wrong input (e.g. 2, a, *, etc.), it asks user to enter a valid input.



HB

Generating Random Speed

After, the user enters the valid input for the length of one section for the zig zag, random speed is generated. This flowchart allows to achieve, the **Functional Requirement 8.**



The reason why I want random number to be less than 15 is because the minimum section length is 15. **Meaning, in the case, when user enters the length of 15 and random number generated is 15 or more, then, either finch will not move at all or move very less.**

For example: time = distance/ speed

time= 15/20, time= 0.75s or 75ms.

On the other hand, the reason why I don't want finch speed to be less than 5, is because as **investigated finch doesn't move very well with speed below 5cm/s (50mm/s).**

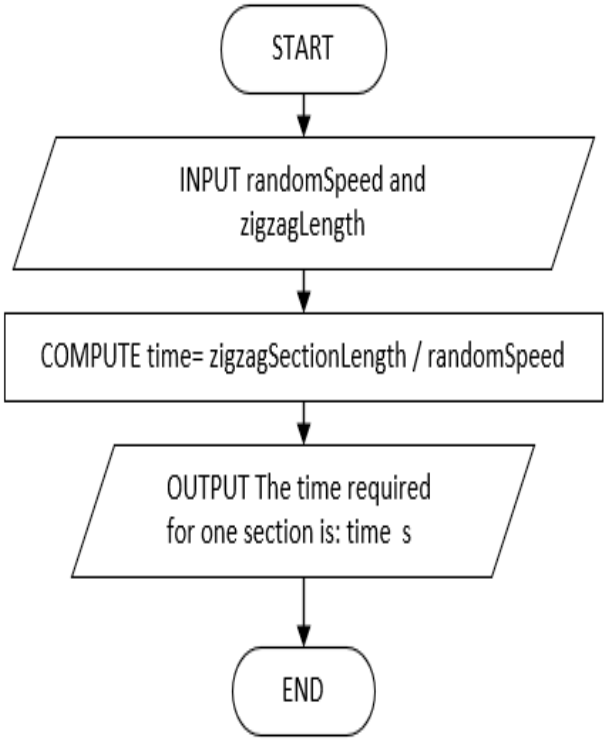
Calculating Time

The flowchart uses the zigzagSectionLength and randomSpeed to calculate time required for the finch to cover distance (section length, which is input by user).

$$Time = Distance \div Speed$$

Note: *this flowchart only calculates the time, it doesn't assign it to finch.*

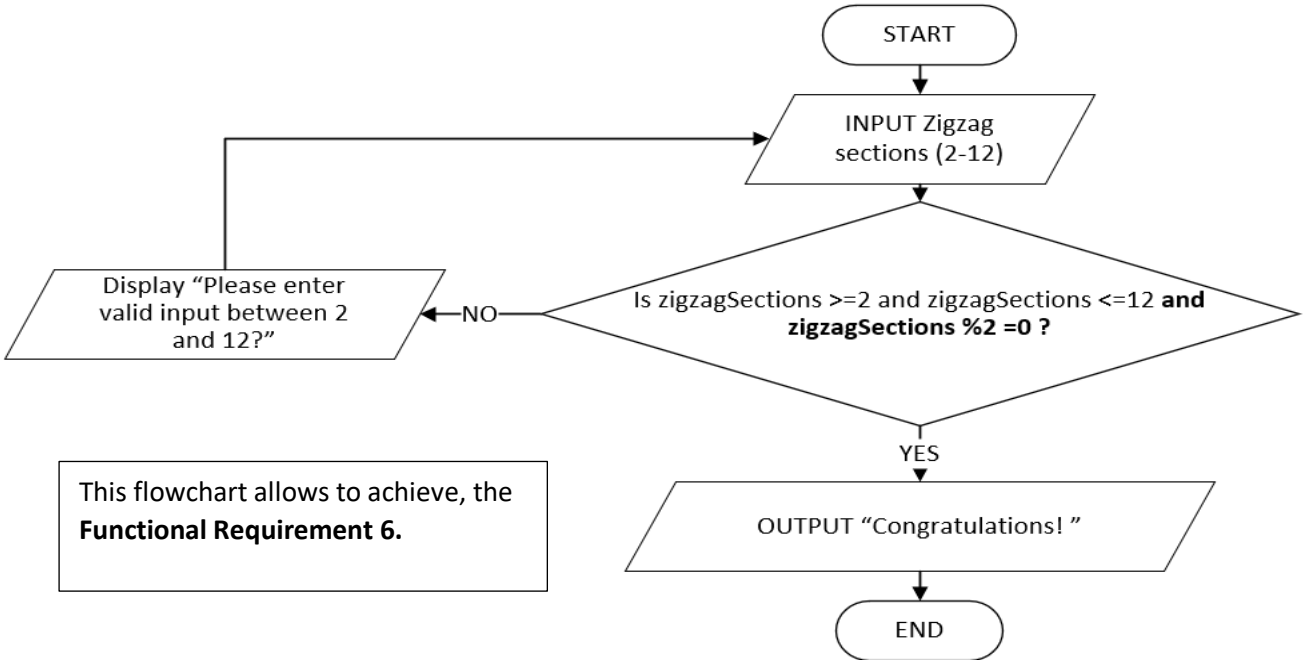
This flowchart allows to attain, the **Functional Requirement 9.**



HB

Input Regular Zigzag Sections

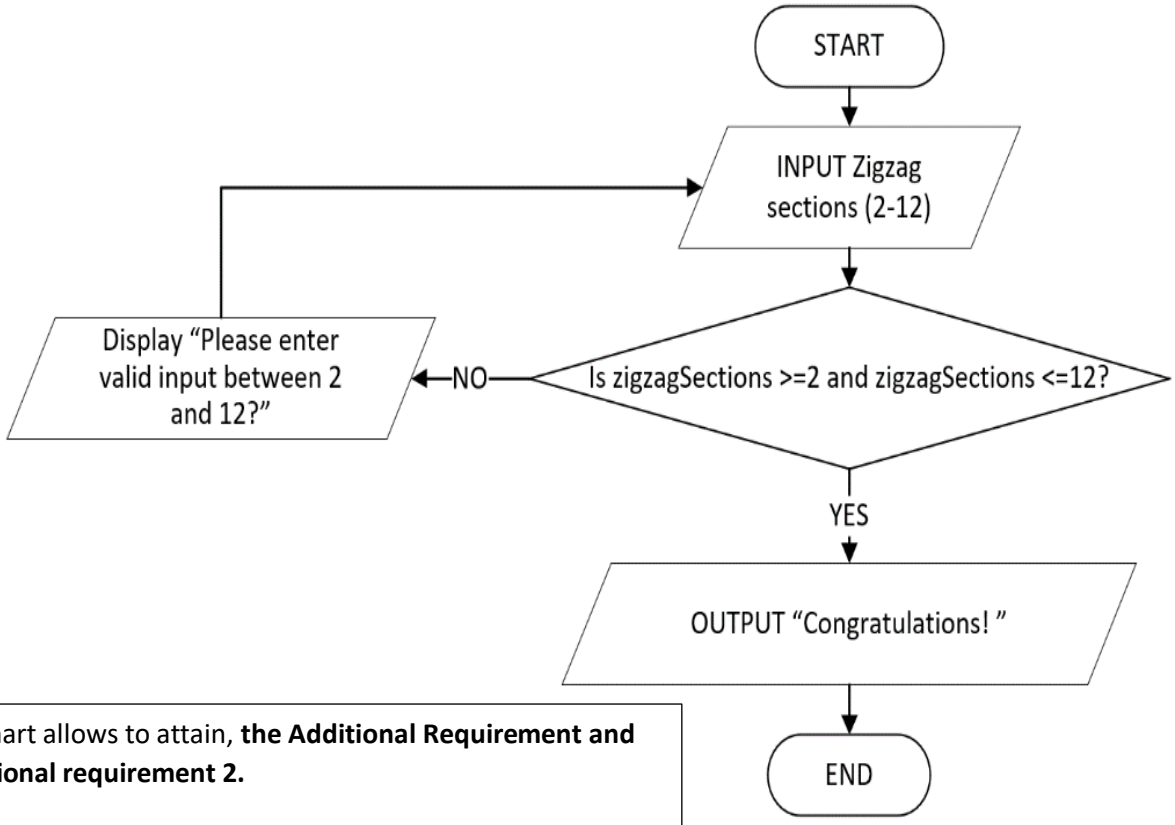
This flowchart ensures, that the user can **only enter the input which is between 2 - 12 and is only an even number.**



This flowchart allows to achieve, the **Functional Requirement 6.**

Input Zigzag Sections

This flowchart is part of freestyle mode, so the user **is not only restricted to even numbers**, they can also input odd numbers of zig zag sections.



This flowchart allows to attain, **the Additional Requirement and Non-Functional requirement 2.**

HB

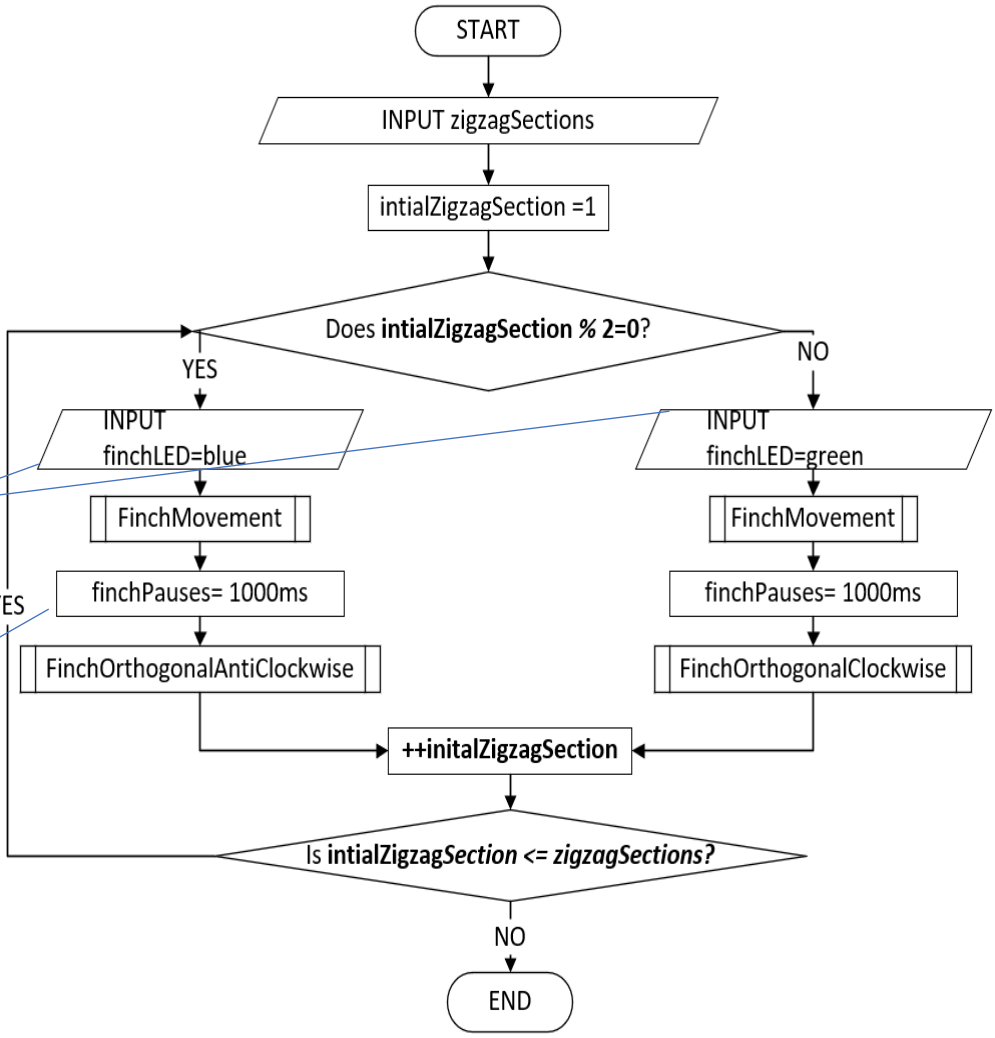
Zigzag Movement

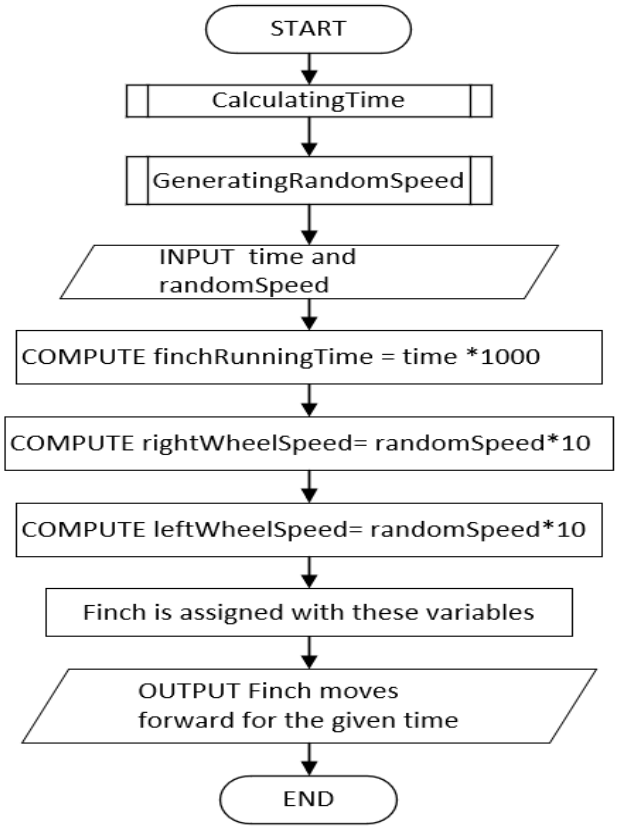
This subprocess consists multiple subprocesses (with their own different functionalities) that collectively come together to make the finch move in zig zag form.

This flowchart allows to achieve, the **Functional Requirements 2,3,4 10 and 11.**

Note: these two inputs allow finch to alternate lights from green to blue and back-forth.

Note: this makes finch pause for 1 second, before it turns at right angle.





Finch Movement

This subprocess, allows the finch to cover the distance input by user and the pauses after the given distance is covered. This subprocess, makes use of two other subprocesses: Calculating time and generating Random speed.

This flowchart allows to achieve, the **Functional Requirement 10.**

*Note: time and speed are converted into **suitable units for finch** (**discovered before as part of Understanding finch**), before they are assigned.*

Which are:

Time= ms (milliseconds)

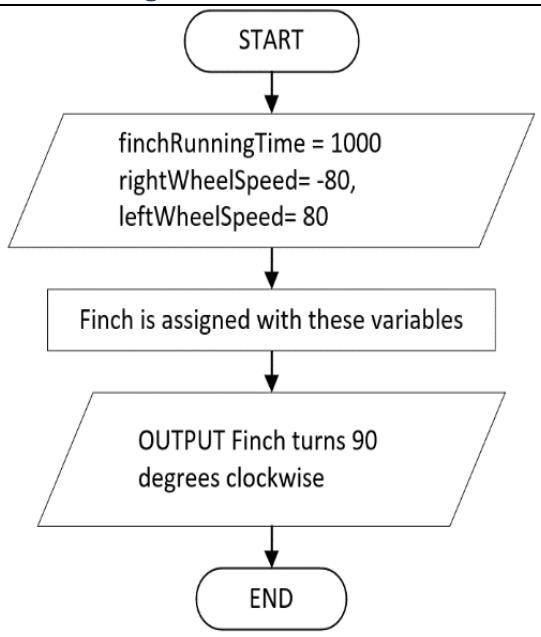
Speed = mm/s(millimetre/seconds)

HB

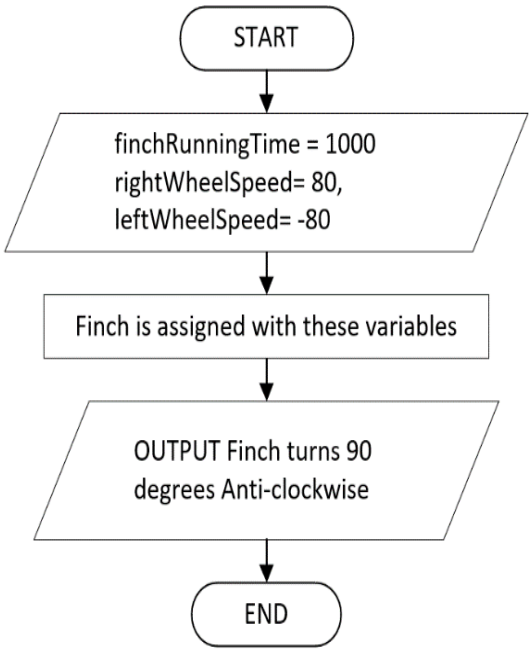
Turning Finch

These flowcharts are responsible for making finch turn at 90 degrees. There are two flowcharts because one makes the finch turn 90 degrees clockwise, whereas, the other one makes it turn anticlockwise. This flowchart allows to achieve, the **Functional Requirements 2 and 4.**

Finch Orthogonal clockwise

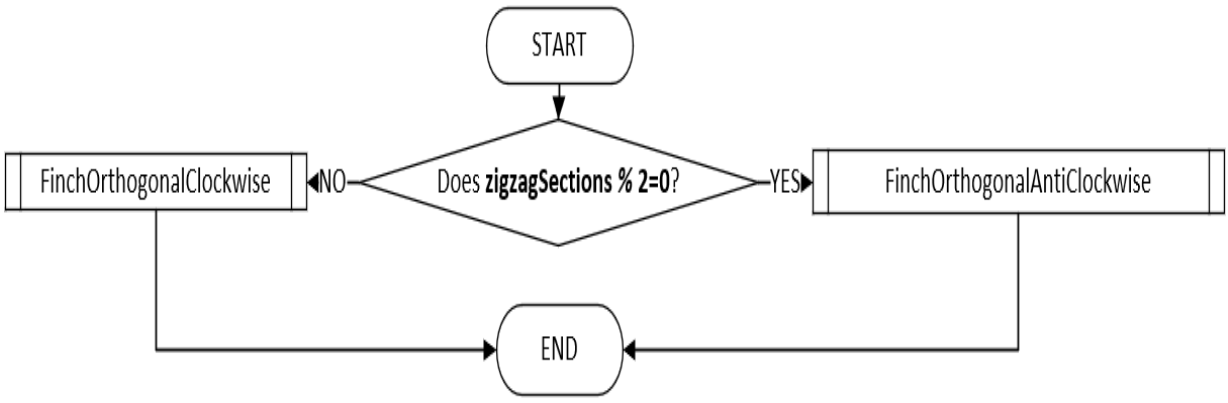


Finch Orthogonal Anticlockwise

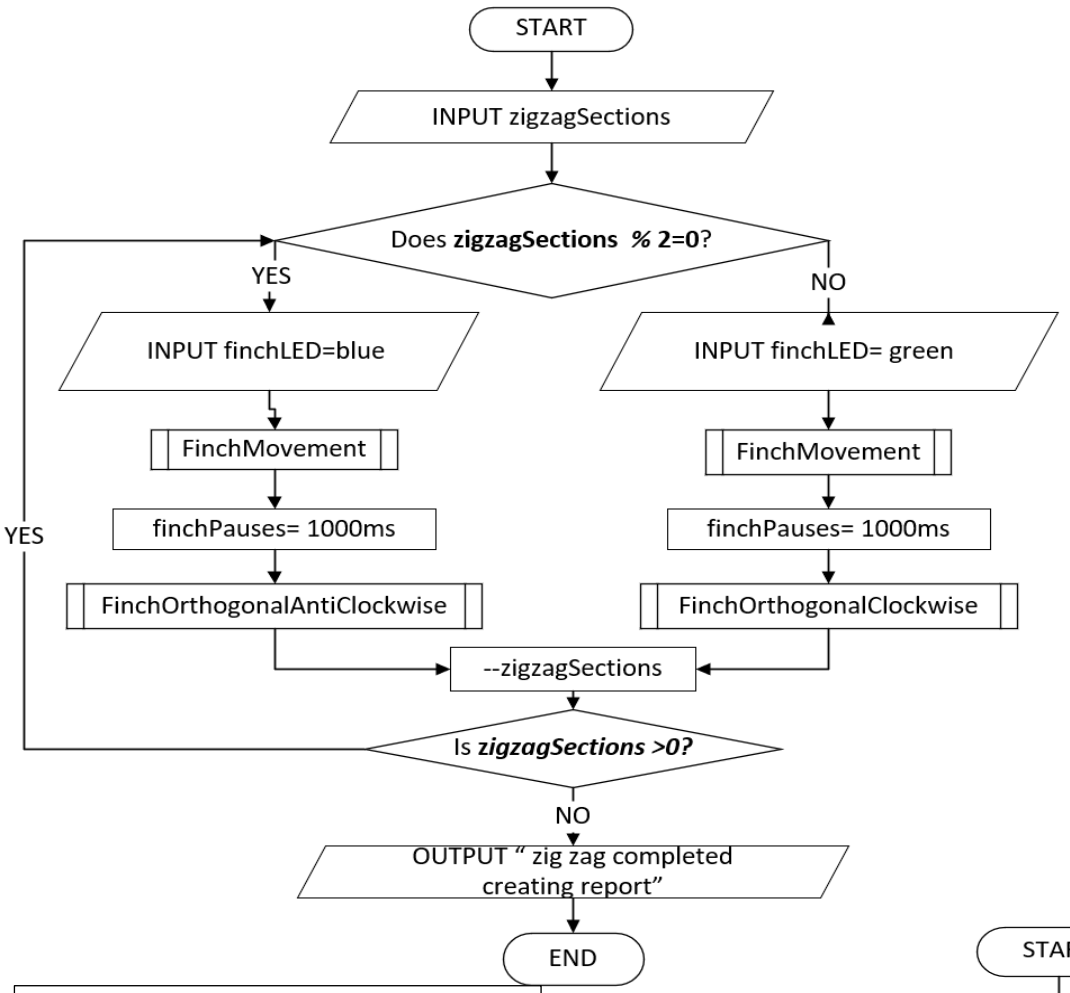


Freestyle turn

This flowchart allows to achieve, the **Additional Functional Requirements.** It turns the finch clockwise or anticlockwise depending on the last section of the finch. **This flowchart makes retracing possible with the freestyle mode**. **For example:** *if the total zigzag section equals to an odd number, after the last section finch turns clockwise (90) then due to this flowchart, it should turn clockwise(90) again, so that, the finch can place itself at a position that can start retracing.*



HB



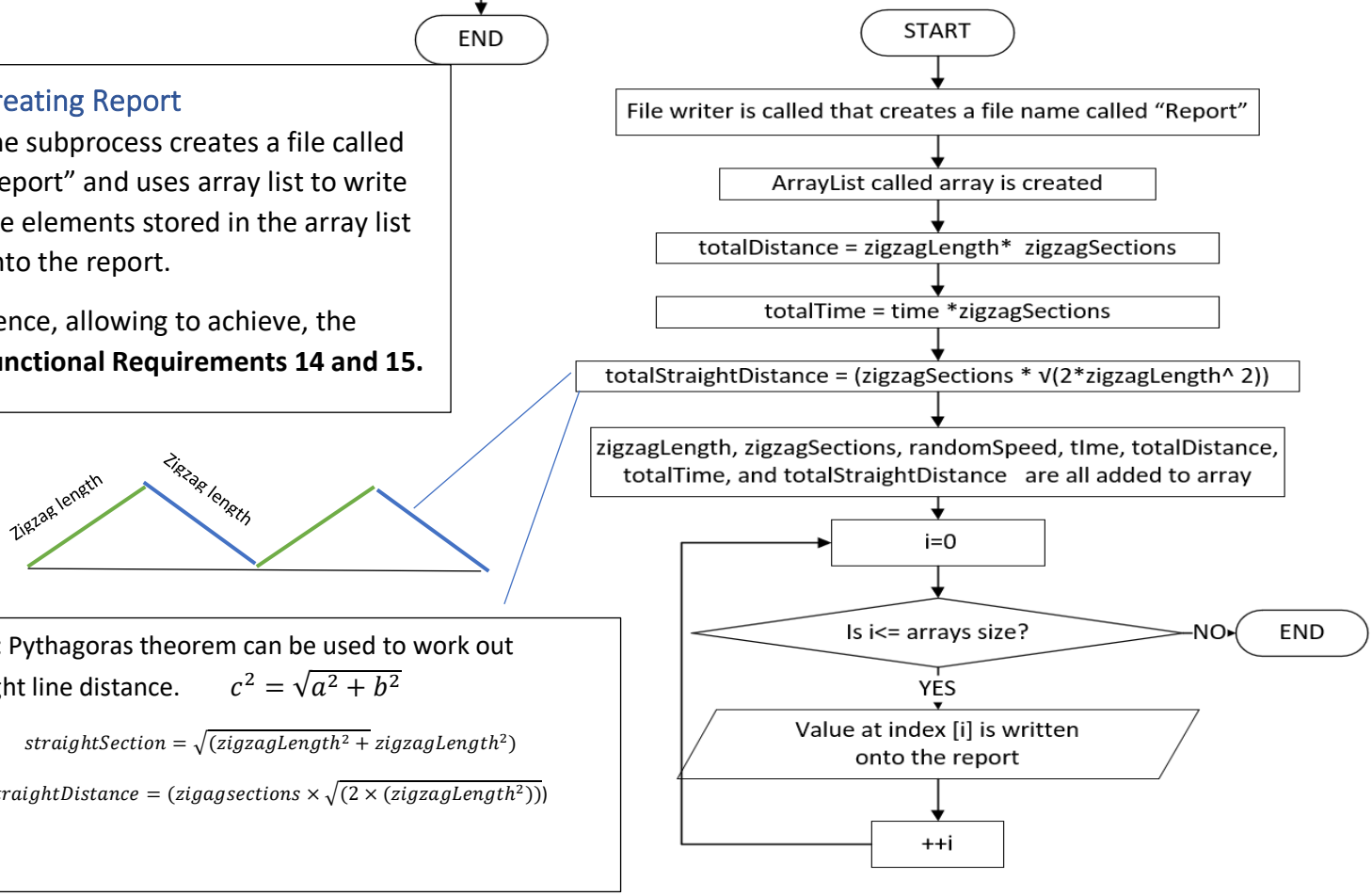
Retrace zig zag

This flowchart allows finch to retrace its steps, this flowchart can be considered as, opposite of zig-zag movement flowchart. This flowchart allows to achieve, the **Functional Requirements 12 and 13.**

Creating Report

The subprocess creates a file called "report" and uses array list to write the elements stored in the array list onto the report.

Hence, allowing to achieve, the **Functional Requirements 14 and 15.**



Note: Pythagoras theorem can be used to work out straight line distance. $c^2 = \sqrt{a^2 + b^2}$

$$straightSection = \sqrt{(zigzagLength^2 + zigzagLength^2)}$$

$$totalStraightDistance = (zigagsections \times \sqrt{(2 \times (zigzagLength^2))})$$

HB

Pseudocode

Algorithm1 Main()

INPUT: NONE

```
1. SUBPROCESS 1 PlaceFincOnFlatSurface
2. SUBPROCESS 2 InputZigzagSectionLength
3. SUBPROCESS 3 GeneratingRandomSpeed
4. SUBPROCESS 4 CalculatingTime
5. OUTPUT "Choose Mode: Regular mode or Freestyle mode"
6. IF the mode = Regular THEN
7.            SUBPROCESS 5 InputRegularZigzagSections
8.            SUBPROCESS 7 ZigzagMovement
9.            SUBPROCESS10 FinchOrthogonalAntiClockwise
10.           SUBPROCESS 12 Retrace
11.           SUBPROCESS 13 CreatingReport
   ELSE
12.           SUBPROCESS 6 InputZigzagSections
13.           SUBPROCESS 7 ZigzagMovement
14.          SUBPROCESS11: FreeStyleTurn
15.           SUBPROCESS 12 Retrace
16.           SUBPROCESS 13 CreatingReport
   END IF
```

OUTPUT: Program makes the finch to form a zigzag, retrace it and create report.

SUBPROCESS1: PlaceFinchOnFlatSurface(Finch_Level)

INPUT: FinchLevel is Boolean data type value, that returns true when finch is on flat surface

```
1. WHILE Finch_Level is NOT True
2.       DISPLAY "Put me on flat service!"
3. END WHILE
```

OUTPUT: "Flat surface detected!"

SUBPROCESS2: InputZigzagLength(zigagSectionLength)

INPUT: User INPUTS zigzagLength is a positive integer

```
1. WHILE zigzagSectionLength is NOT >= 15 and zigzagSectionLength
   is NOT <= 85 THEN
2.            "Please enter valid input between 15 and 85"
3. END WHILE
```

Output: DISPLAY "Congratulations!"

HB

SUBPROCESS3 GenrateRandomSpeed()

INPUT: None

1. **WHILE** randomNumber is **NOT** >5 **AND** randomNumber is **NOT** <15
2. **GENERATE** a randomNumber
3. **END WHILE**
4. randomSpeed = randomNumber

Output: **DISPLAY "speed = randomSpeed cm/s"**

SUBPROCESS4: CalculatingTime(zigzagSectionLength, randomSpeed)

INPUT: zigSectionzagLength and randomSpeed are positive integers.

1. **COMPUTE** time= zigzagSectionLength ÷ randomSpeed

Output: DISPLAY "The Time required for one section is: **time** s"

SUBPROCESS5: InputRegularZigzagSections(zigzagSections)

INPUT: zigzagSections is a positive integer, that the user inputs

1. **WHILE** ((zigzagSections is **NOT** >= 2 and zigzagSections is **NOT** <= 12) and (zigzagSection modulo 2 **is NOT=** 0))
2. "Please enter valid input between 2 and 12"
3. **END WHILE**

OUTPUT: Display "Congratulations!"

SUBPROCESS6: InputZigzagSections(zigzagSection)

INPUT: zigzagSections is a positive integer, that the user inputs

1. **WHILE** zigzagSections is **NOT** >= 2 and zigzagSections is **NOT** <= 12
2. "Please enter valid input between 2 and 12"
3. **END WHILE**

OUTPUT: Display "Congratulations!"

HB

SUBPROCESS7: ZigzagMovement(zigzagSections)

INPUT: zigzagSections is a positive integer, that the user inputs

```
1. intialZigzagSection= 1
2. WHILE intialZigzagsection <=ZigzagSections
3.            IF intialZigzagSection %2=0 THEN
4.                  OUTPUT finchLED= blueColor
5.                  SUBPROCESS 8 FinchMovement
6.                  finchPauses = 1000ms
7.                  SUBPROCESS 10 FinchOrthogonalAntiClockwise
8.                  ++intialZigzagSection
              ELSE
9.                  OUTPUT finchLED= greenColor
10.                 blueColor = 0
11.                 SUBPROCESS 8 FinchMovement
12.                 finchPauses = 1000ms
13.                 SUBPROCESS 9FinchOrthogonalClockwise
14.                 ++intialZigzagSection
              END IF
   END WHILE
```

OUTPUT: Finch moves in a zigzag manner

SUBPROCESS8: FinchMovement(time, randomSpeed)

INPUT: time and randomSpeed are positive integers.

1. **SUBPROCESS4** CalculatingTime
2. **SUBPROCESS5** GenrateRandomSpeed
3. **COMPUTE** finchRunningTime= time * 1000
4. **COMPUTE** rightWheelSpeed= randomSpeed * 10
5. **COMPUTE** leftWheelSpeed= randomSpeed * 10
6. time, rightWheelSpeed and leftWheelSpeed are assigned to finch.

Output: Finch moves for the given time at random speed

SUBPROCESS9: FinchOrthogonalClockwise(experimentResultTime, rightWheelSpeed, leftWheelSpeed)

INPUT: experimentResultTime and leftWheelSpeed are positive integers. Whereas, rightWheelSpeed is negative integer.

1. experimentResultTime = 1000
2. rightWheelSpeed= -80
3. leftWheelSpeed= 80
4. Finch_Set_velocity = experimentResultTime, leftWheelSpeed and rightWheelSpeed

Output: Finch turns 90 degrees clockwise.

SUBPROCESS10: FinchOrthogonalAntiClockwise(experimentResultTime, rightWheelSpeed, leftWheelSpeed)

INPUT: experimentResultTime and rightWheelSpeed are positive integer. Whereas, leftWheelSpeed is negative integer.

1. experimentResultTime = 1000
2. rightWheelSpeed= 80
3. leftWheelSpeed= -80
4. Finch_Set_velocity = experimentResultTime, leftWheelSpeed and rightWheelSpeed

Output: Finch turns 90 degrees anticlockwise

SUBPROCESS11: FreeStyleTurn(zigzagSections)

INPUT: zigzagSections is a positive integer

1. **IF** zigzagSections %2=0 **THEN**
2. **SUBPROCESS 10**FinchOrthogonalAntiClockwise
3. **ELSE**
4. **SUBPROCESS 9** FinchOrthogonalClockwise
5. **END IF**

OUTPUT: NONE

SUBPROCESS12: RetraceZigzag(zigzagSectionLength)

INPUT: zigzagSection is a positive integer

1. backwardsZigzagSection= zigzagSection
2. **WHILE** backwardsZigzagSection >0
3. **IF** backwardsZigzagSection %2=0 **THEN**
4. OUTPUT finchLED= blueColor
5. **SUBPROCESS 8**FinchMovement
6. finchPauses = 1000ms
7. **SUBPROCESS 10** FinchOrthogonalAntiClockwise
8. **--intialZigzagSection**
   ELSE
9. OUTPUT finchLED= greenColor
10. **SUBPROCESS 8**FinchMovement
11. finchPauses = 1000ms
12. **SUBPROCESS 9**FinchOrthogonalClockwise
13. **--intialZigzagSection**
    END IF
    END WHILE

OUTPUT: "Creating report"

HB

SUBPROCESS13: CreatingReport(zigzagLength, totalZigzagSection, speedInCMPersecond, timeInSeconds)

INPUT: zigzagLength, totalZigzagSection, speedInCMPersecond, and timeInSeconds are all positive integers

1. File Writer creates a file called "report" is created in temp folder in h drive
2. Array List called array is created
3. **COMPUTE traversedPathLength = zigzagLength * totalZigzagSection**
4. **COMPUTE totalTimeForCompletion= (timeInSeconds* totalZigzagSection)+(totalZigzagSection)**
5. **COMPUTE** $\boldsymbol{totalStraightDistance} = (\boldsymbol{zigagsections} \times \sqrt{(2 \times (\boldsymbol{zigzagLength}^2))})$
6. zigzagLength, totalZigzagSection, speedInCMPersecond, timeInSeconds, traversedPathLength, totalTimeForCompletion and totalStraightDistance are all converted into Strings and added to array
7. **FOR i=0**
8. **If** i<= array's size
9. Add array[i] onto "report"
10. INCREMENT i
11. **END IF**

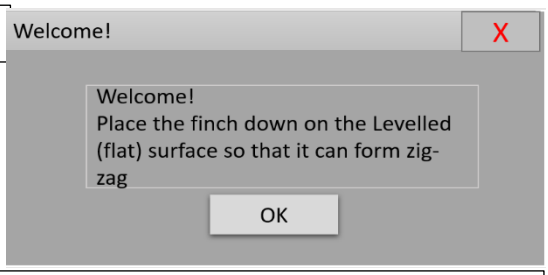
END FOR

Output: "Report is created!"

User interface prototype

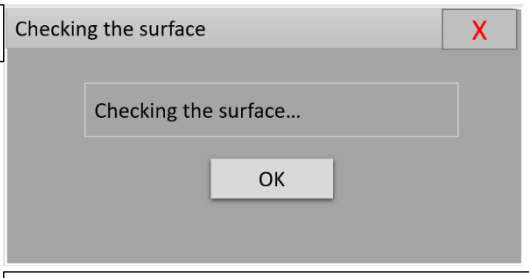
Designing user-Interface is an important part of software design as **it gives an insight on how the program should behave, before implementation**. I've taken into consideration, everyone's lack of familiarity with this task. Therefore, I've designed my UI in a way is very simple and it will instruct the user on how the zig-zag program works, thus, allowing me to achieve both functional and non-functional requirements.

1



When the program is first run this screen pops-up, it tells the user, to **put finch down on levelled surface**, so the surface test can be carried out.

2

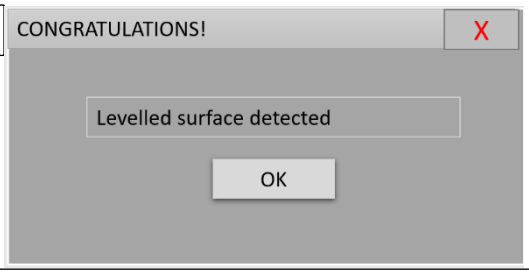


The second screen that pops up, when the user clicks on **'OK'** the finch checks whether it's on Level surface or not.

3

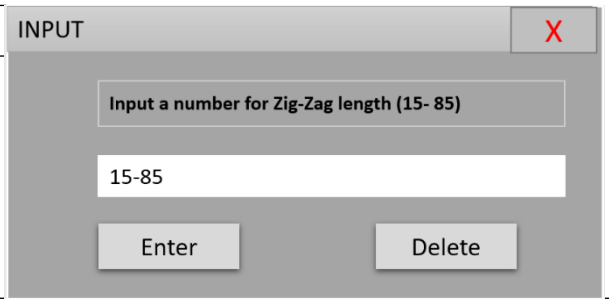
If the finch is not on Levelled surface this dialog box pops up

4



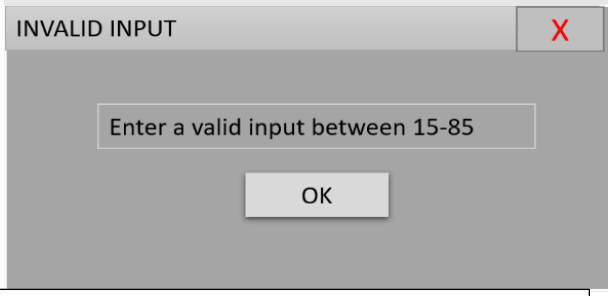
If it is on levelled surface, then this dialog box pops up. **When User clicks on 'OK' it takes the user to Input Zig Zag Length window.**

5



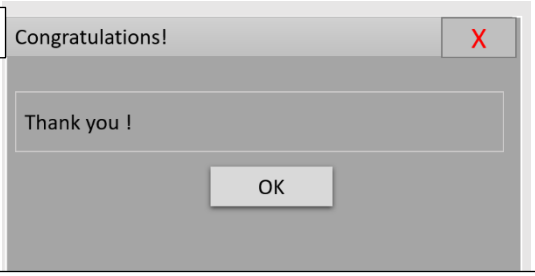
This screen asks the user to enter the number between 15-85 for zig zag length. **The "ok" buttons allows to enter input, however, the "Delete" allows to delete the input number**.

6



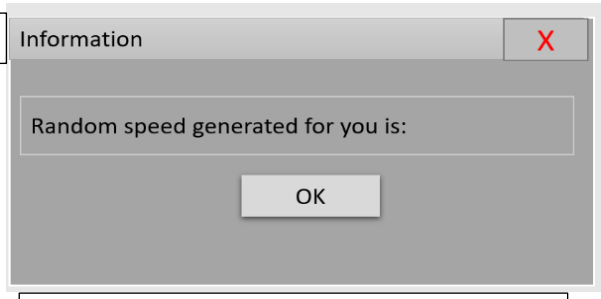
If the user enters wrong Input such as number out of limit, this screen pops up. **When user clicks on 'OK' they can renter again.**

7



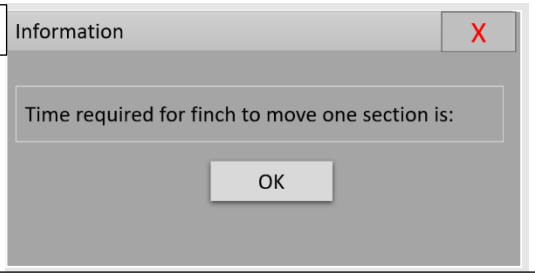
When User enters valid input, this screen pops up. **When user presses 'OK' random speed is generated**.

8



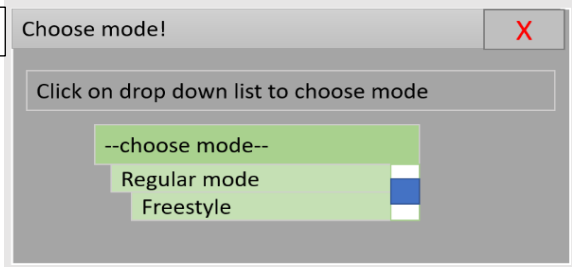
This screen tells the user the random speed that is generated. **When user presses 'OK' time is calculated.**

9



This screen tells the user how long the finch will move for to cover the distance, earlier input by user. **When user clicks on 'OK' the mode screen pops up.**

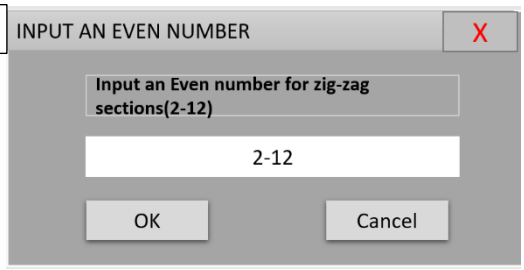
10



This screen shows different mode of zig zag. It shows drop down menu where the user can select the desirable mode.

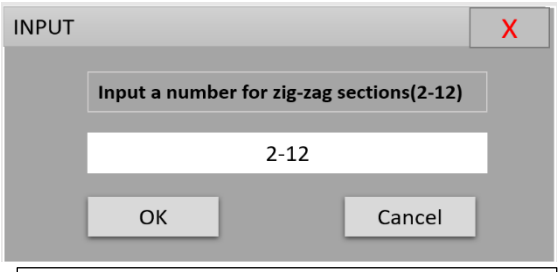
HB

11



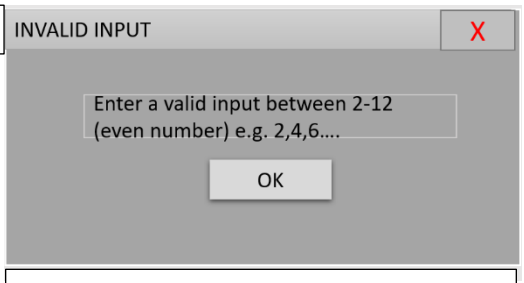
If the user **chooses regular mode**, this screen pops up where they could only input even number.

12



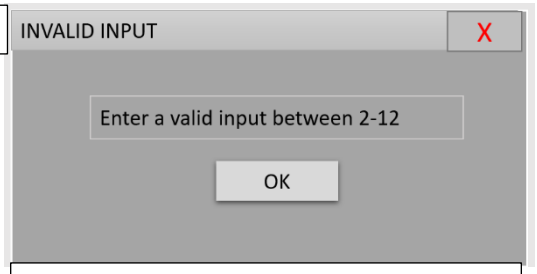
If the user **chooses freestyle mode**, this screen pops up where they could any number.

13



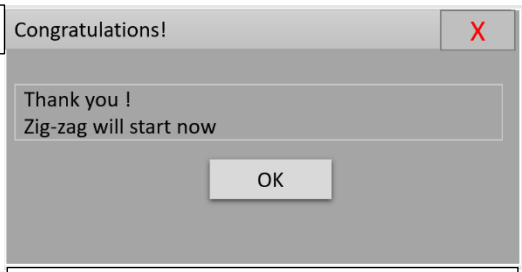
If user **enters invalid input** in regular mode, this screen pops up.

14



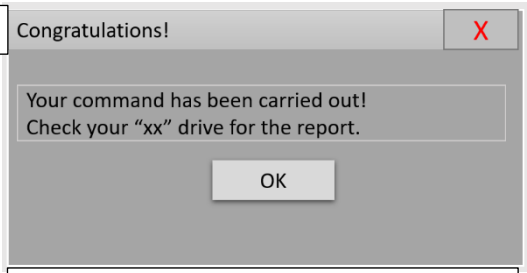
If user **enters invalid input** in freestyle mode, this screen pops

15



If user **enters valid input** on any mode, this screen pops up.

16



After finch has stopped traversing, this screen pops up.

17

If on any of the message pane the 'X' button is clicked on, the program will end.

HB