



CHIANG MAI UNIVERSITY
College of Arts, Media and Technology
1st Semester / Academic Year 2025

960101 Fundamentals of Programming Logic in Digital Industry

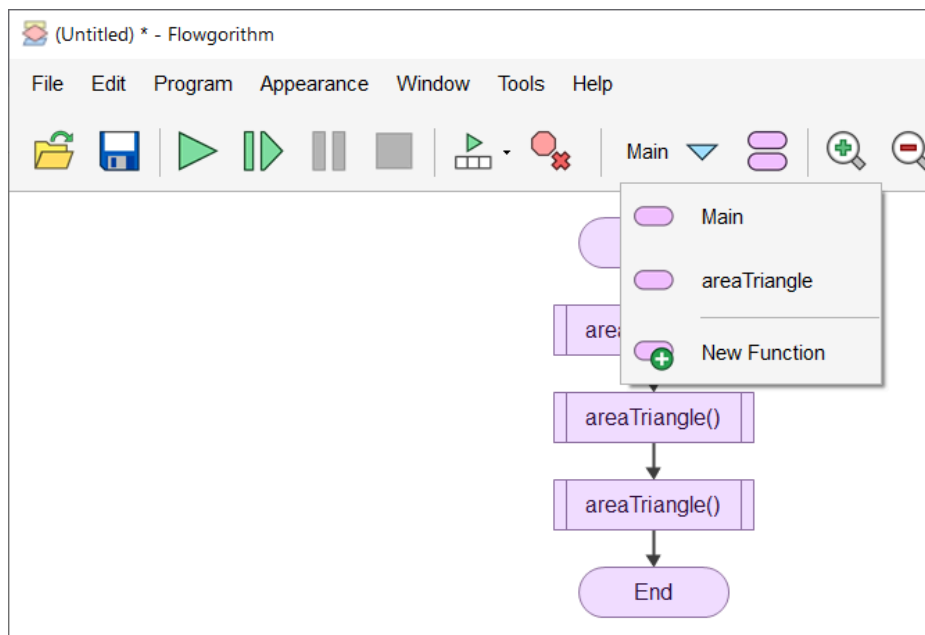
Lab Assignment 11: Function

Name Student ID Section.....

Objectives:

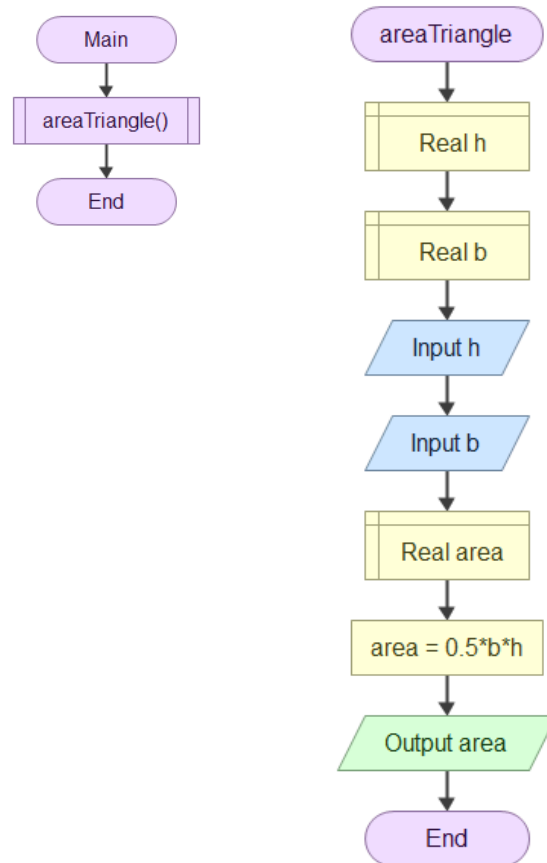
- 1) Understand the concept of functions.
- 2) Be able to create a function in Flowgorithm and Thinkable.
- 3) Be able to pass parameters to the created function, to process something inside the function, and to return a value.

Part 1 – Flowgorithm Tutorial

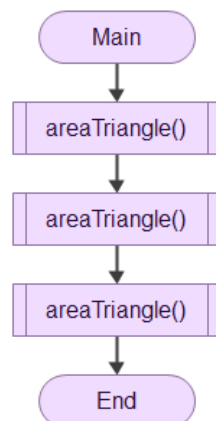


Ex. 1.1 Function without return value and without parameter.

1.1.1 Create a flowchart to calculate the area of triangle while inputs are height and base of triangle using function without returning and without parameter (areaTriangle()). Then, display the answer.

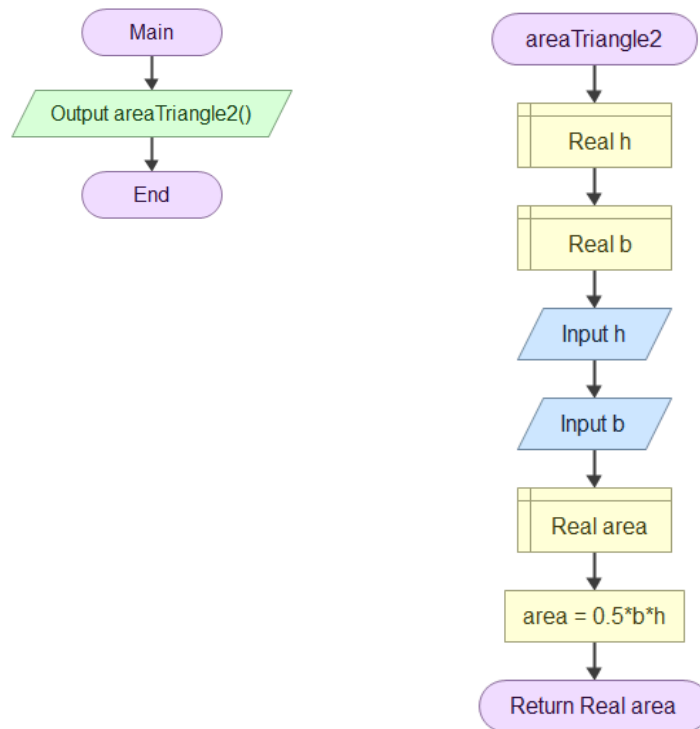


1.1.2 Modify the flowchart to display the area of 3 triangles by calling the **areaTriangle()** function 3 times.



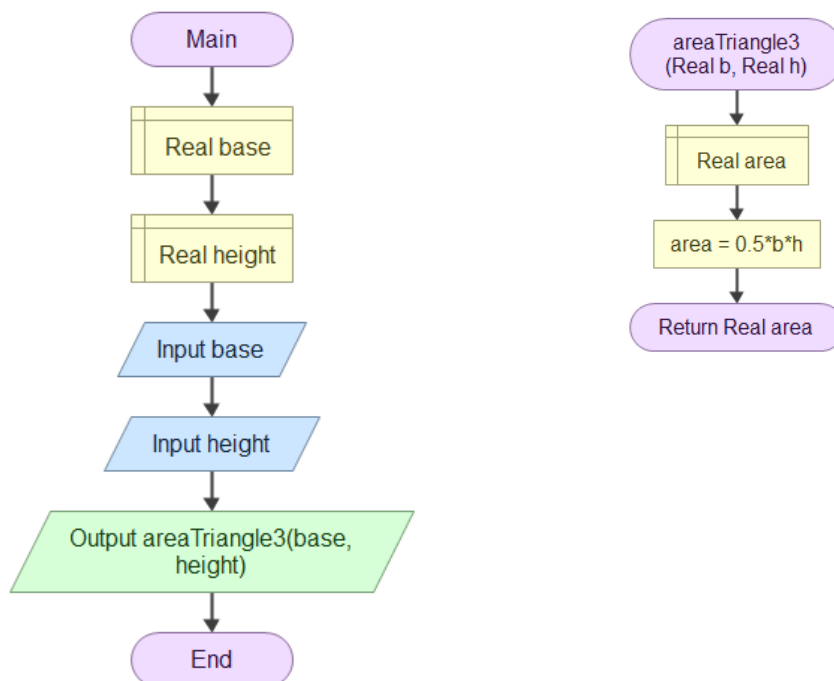
Ex. 1.2 Function with return value and without parameter.

1.2.1 Modify the flowchart from 1.1.2 to use **areaTriangle()** function with returning area of triangle.



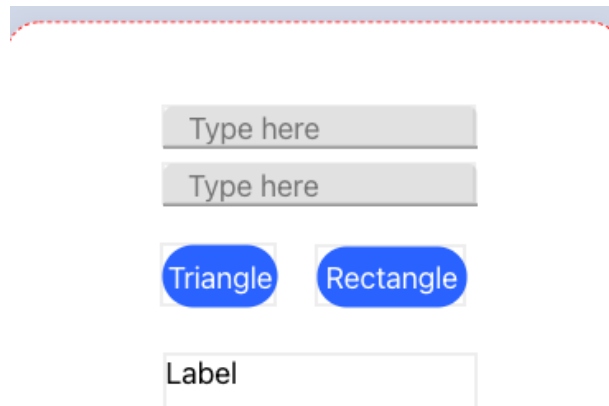
Ex. 1.3 Function with return value and with parameter.

1.3.1 Modify the flowchart from 1.1.2 to use **areaTriangle()** function with returning area of triangle. Then, use height and base of the triangle as the parameter of the function.



Part 2 – Thunkable Tutorial

Create a program to create 2 buttons, 2 textboxes, 1 label. The program will calculate the area of triangle if the button “Triangle” is clicked or calculate the area of rectangle if the button “Rectangle” is clicked. Then, display the answer in the label.



The image shows a Thunkable visual programming interface. At the top, there is a light blue header bar with a red dashed line indicating a sequence of blocks. Below the header, there are five blocks arranged vertically: two 'Type here' text input blocks, two 'Triangle' and 'Rectangle' button blocks, and a 'Label' output block.

Type here

Type here

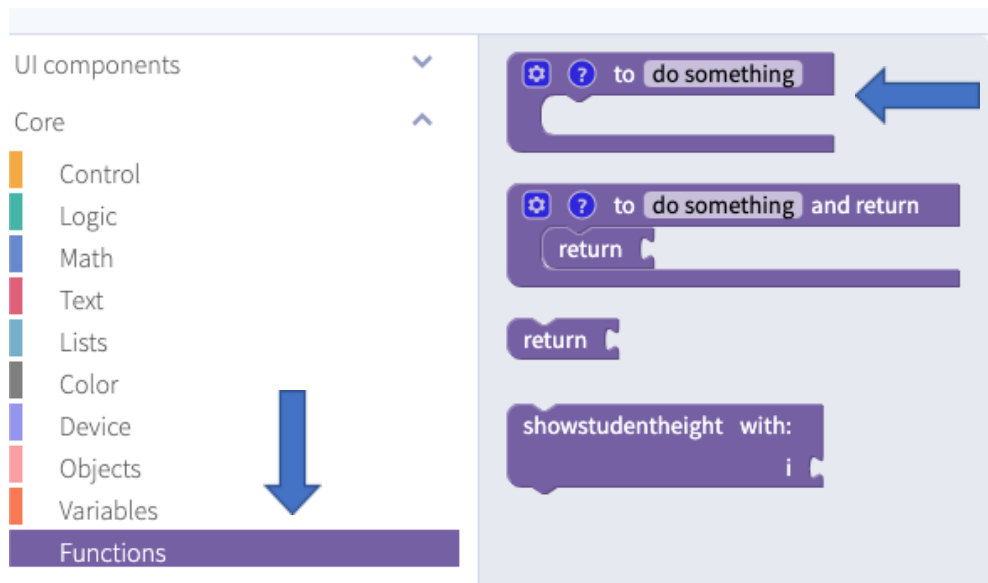
Triangle

Rectangle

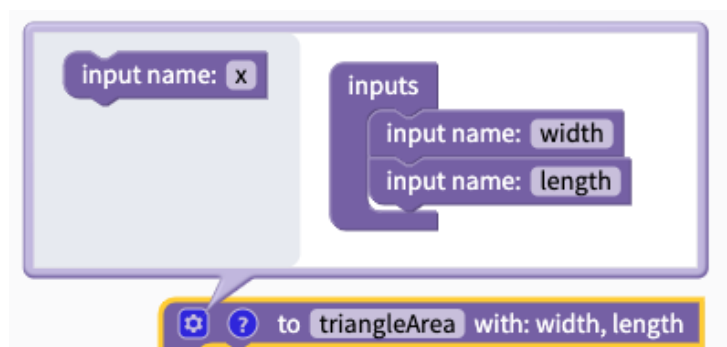
Label

Function without return value

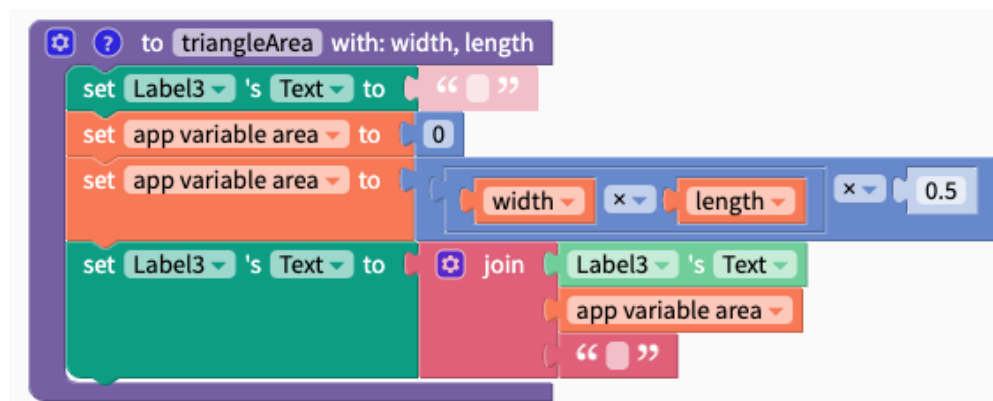
- Create the function without return value



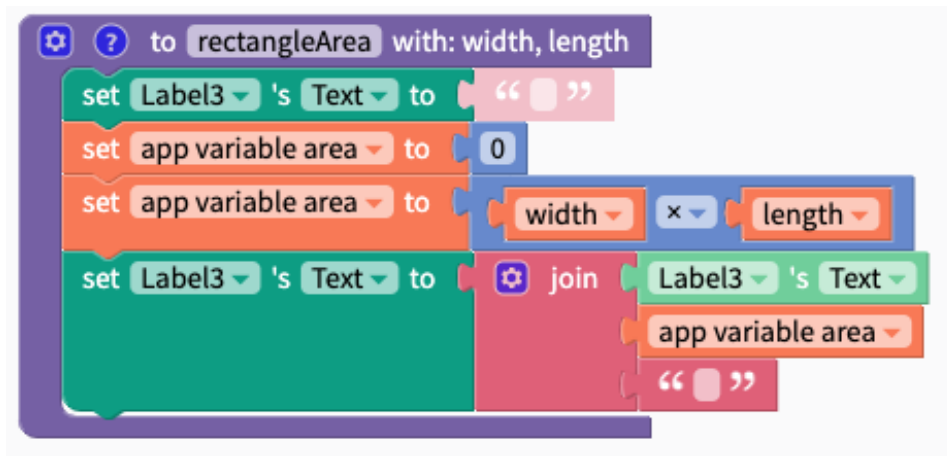
- Specify the input argument by clicking at gear symbol and dragging the input.
- Change the input argument's name



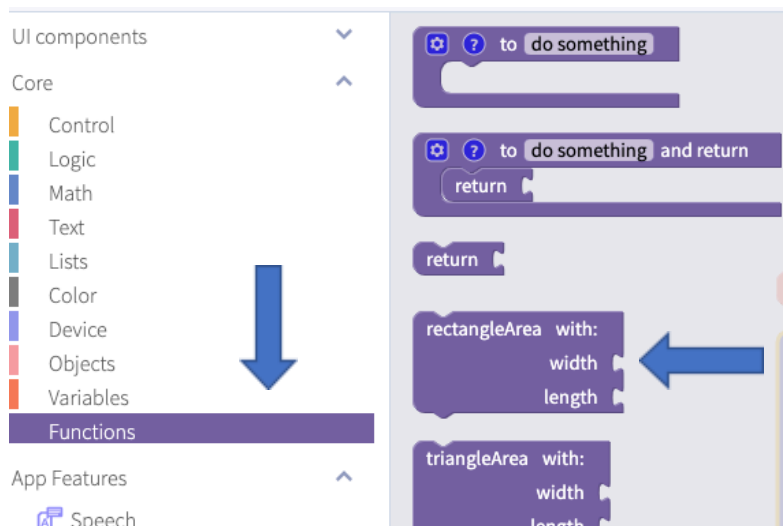
- Create "triangleArea" function



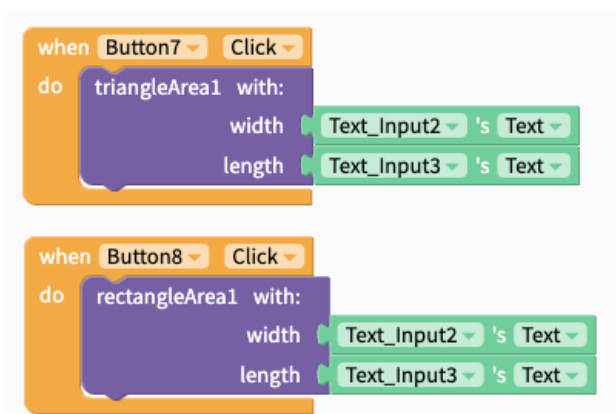
- Create “rectangleArea” function



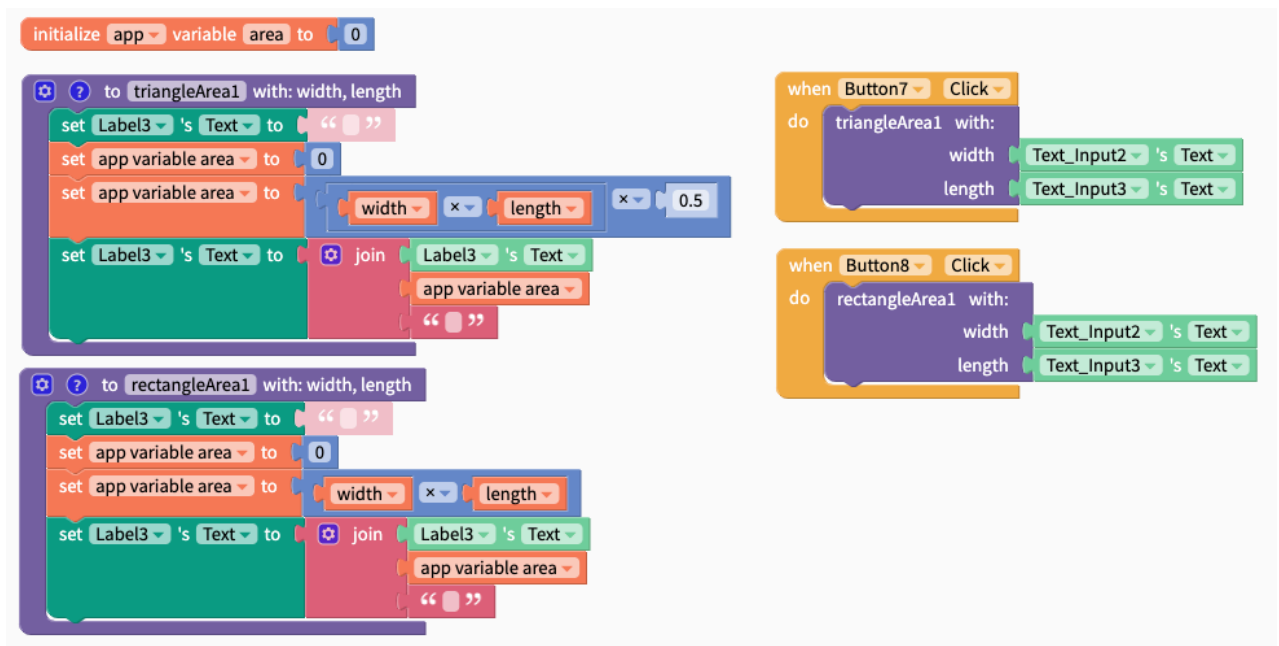
Call function



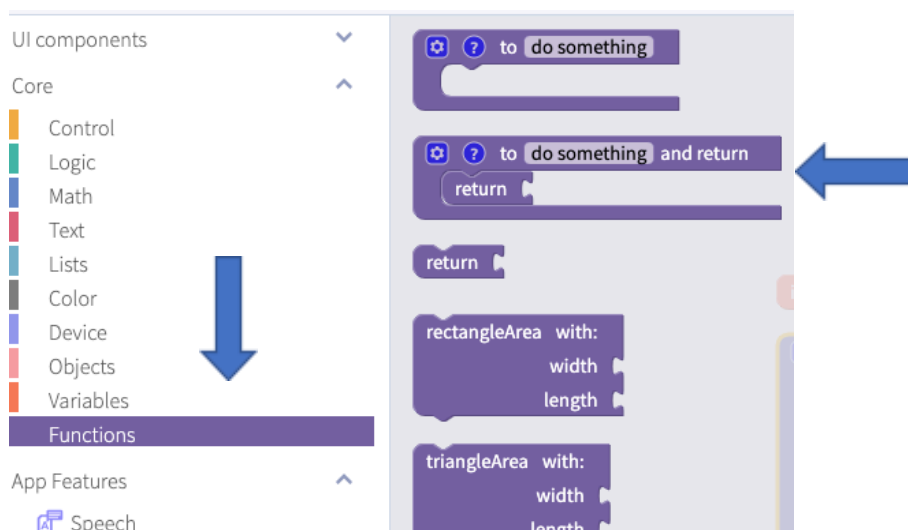
- Call function input the arguments

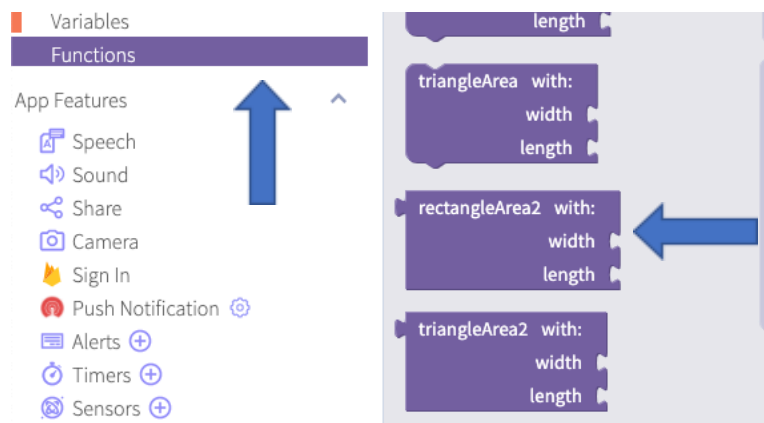


The complete code using function without return value

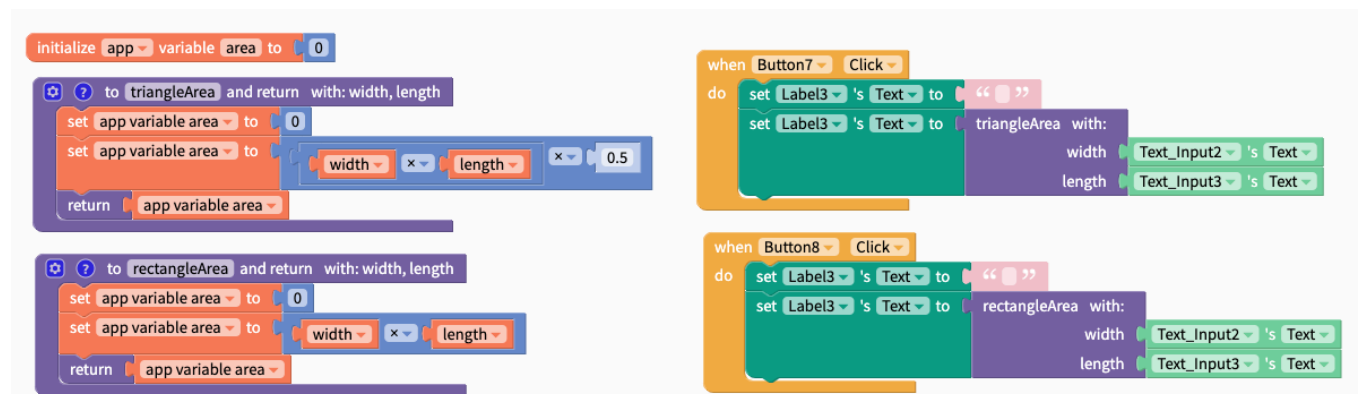


Function with return value





The complete code using function with return value

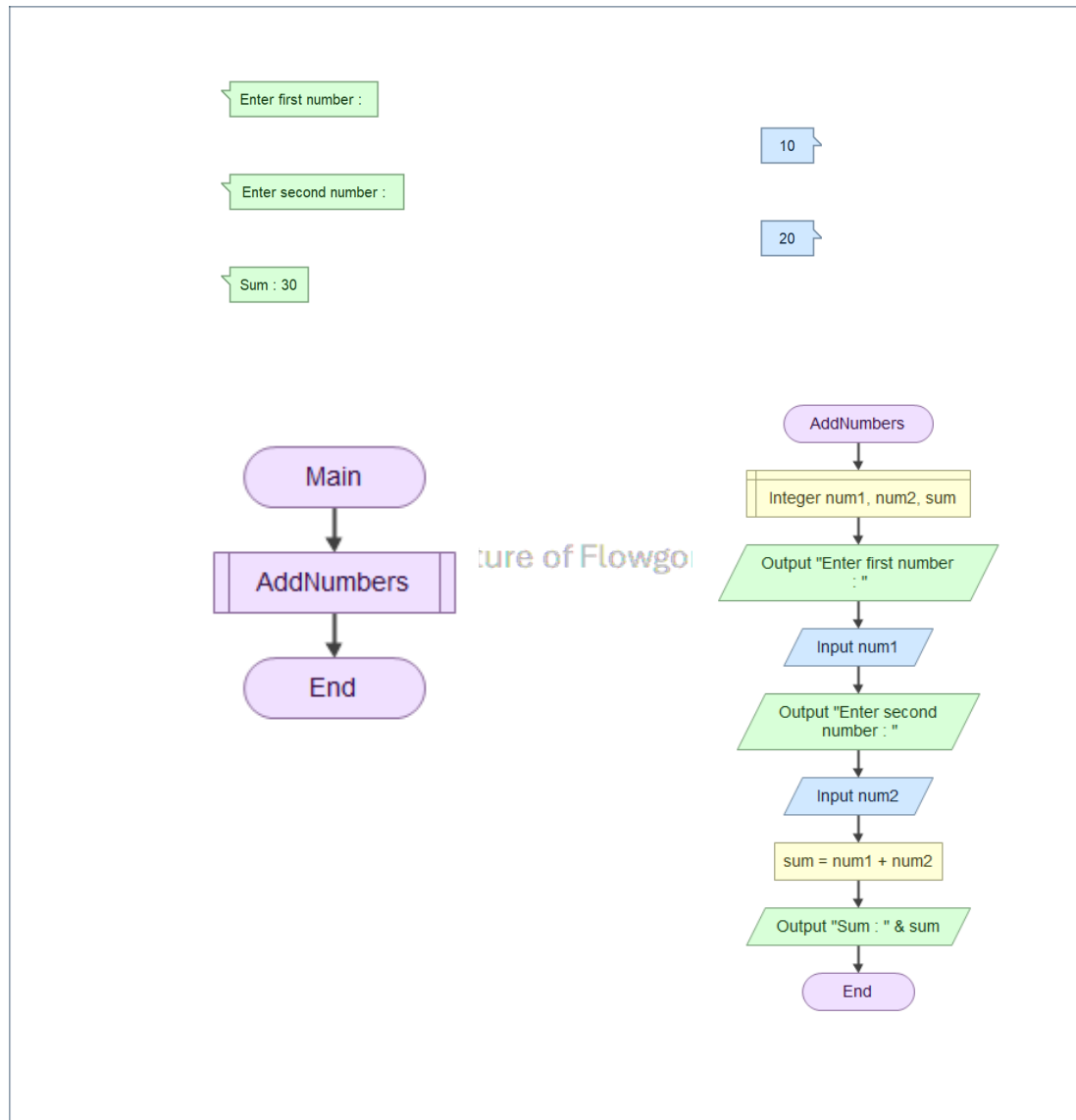


Part 2 – Problem Sets

1. Create a program to receive 2 integer values and return the addition of the inputs. (Use Function without return value and without parameter)

1.1 Create a flowchart on **Flowgorithm**.

Answer:



2.

1.2 Create a program on **Thunkable**.

Answer:

The screenshot displays a Thunkable app interface with three text input fields and a button. The first input field is labeled "First number :" and contains the value "13". The second input field is labeled "Second number :" and contains the value "3". Below these fields, the text "Sum : 16" is displayed. A blue button labeled "Button" is positioned below the sum. Below the interface, the code blocks for the app are shown. The code includes three initialization blocks for variables num1, num2, and sum. A function block named "AddNumbers" contains logic to set the variables from the input fields, calculate the sum, and update the label. This function is triggered by a "when Button1 Click" event.

First number :

13

Second number :

3

Sum : 16

Button

Insert screen capture of Thunkable HERE!!

```
initialize app variable num1 to
initialize app variable num2 to
initialize app variable sum to

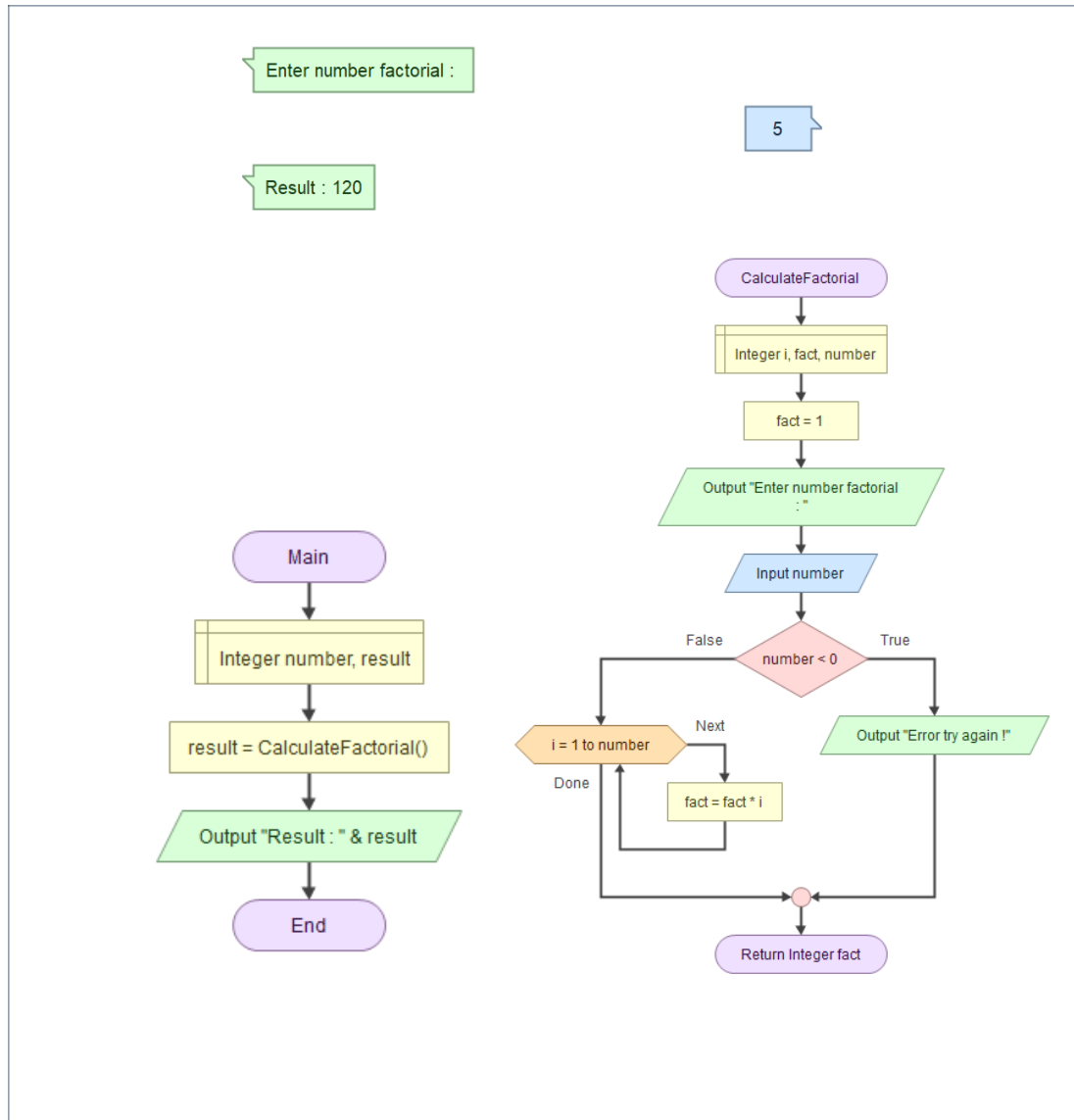
? + to AddNumbers
  set app variable num1 to Text_Input1's Text
  set app variable num2 to Text_Input2's Text
  set app variable sum to app variable num1 + app variable num2
  set Label3's Text to + join - " Sum : " - app variable sum

when Button1 Click
  do AddNumbers
```

2. Create a program to calculate the factorial of input. (**Use Function with return value and without parameter**)

2.1 Create a flowchart on **Flowgorithm**.

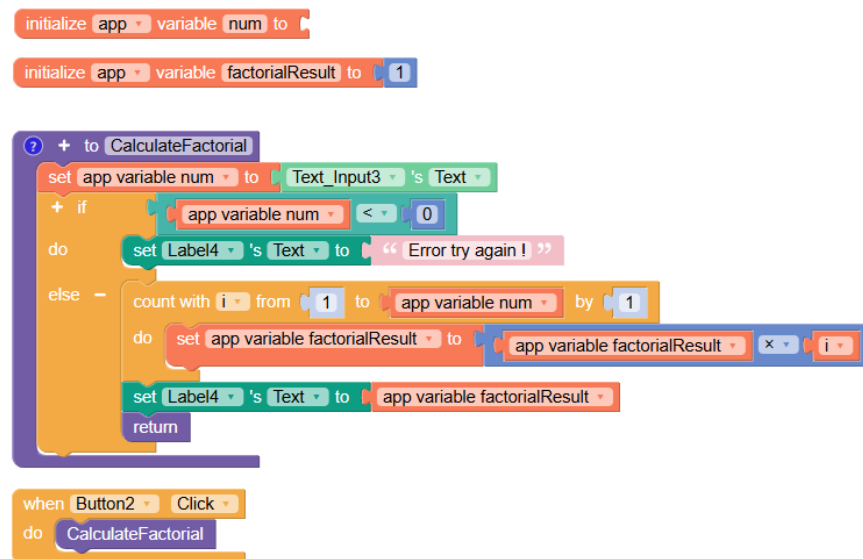
Answer:



3.

2.2 Create a program on **Thunkable**.

Answer:



Insert screen capture of Thunkable HERE!!

6

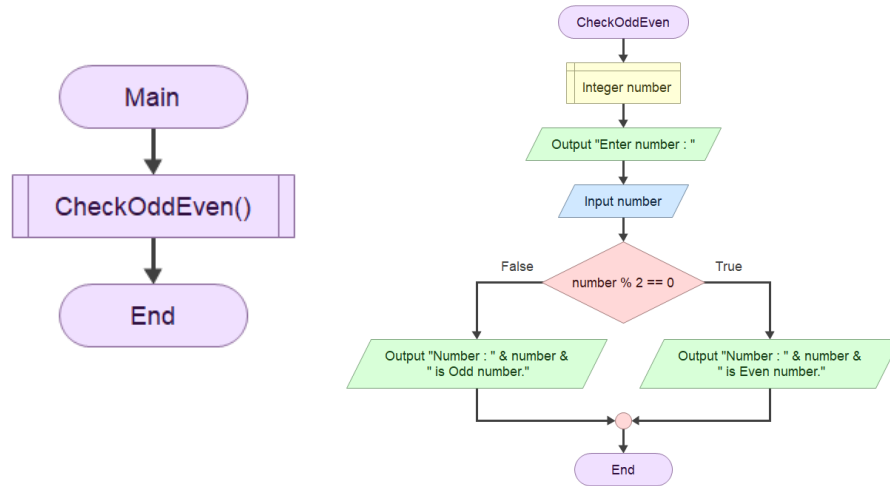
720

Button

3. Create a program to determine if a number is odd number, or not. (Use Function without return value and with parameter)

3.1 Create a flowchart on **Flowgorithm**.

Answer:



Insert screen capture of Flowgorithm HERE!!

Enter number :

Number : 10 is Even number.

10

4.

3.2 Create a program on **Thunkable**.

Answer:

The screenshot displays a Thunkable app interface and its underlying code. The interface at the top features a text input field containing the number "6", a blue button labeled "Button", and a text label below it that reads "Number : 6 is Even number.".

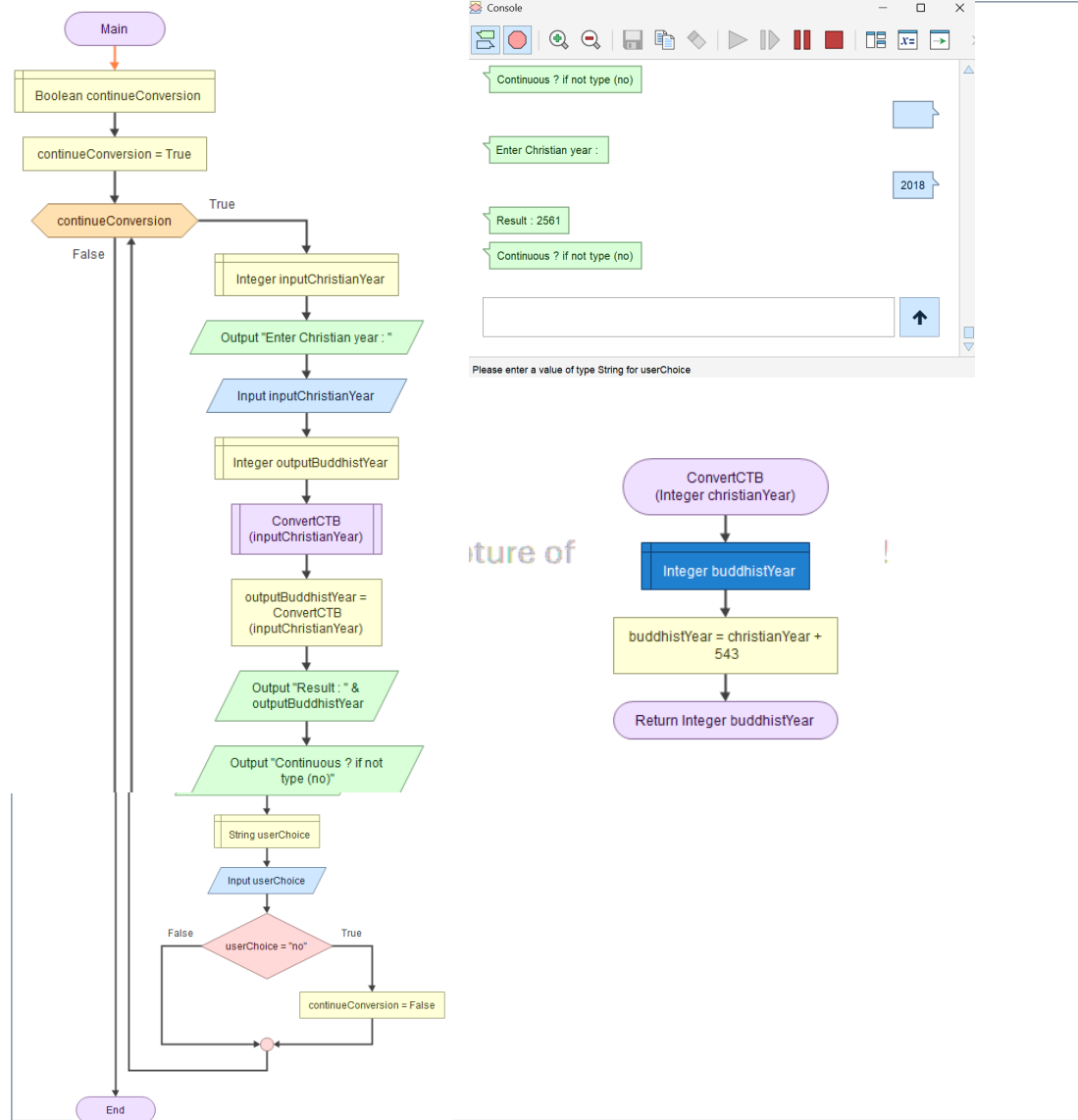
The code section below the interface is organized into three main parts:

- Initialize:** An orange block labeled "initialize app variable number03 to" is positioned at the top of the code area.
- CheckOddEven Function:** A purple block labeled "+ to CheckOddEven" defines a function. Inside this function:
 - A green block "set app variable number03 to" is connected to "Text_Input4's Text".
 - An orange "if" block checks the condition "remainder of app variable number03 ÷ 2 = 0".
 - If the condition is true (do branch), a green "set Label5's Text to" block is connected to a pink "join" block. The join block concatenates the strings "Number: ", the value of "app variable number03", and " is Even number.".
 - If the condition is false (else branch), another green "set Label5's Text to" block is connected to a pink "join" block. This join block concatenates "Number: ", the value of "app variable number03", and " is Odd number.".
- Click Event:** An orange block labeled "when Button3 Click" is connected to the "do" slot of the "CheckOddEven" function block.

4. Create a flowchart on **Flowgorithm with a function** that convert Christian year to Buddhism year. The function must receive a parameter and return an output. The program must be able to convert multiple years at the same time.

Hint: you need to determine inputs, outputs, and the process of the year conversion.
you need to know which parts should be grouped as a function.

Answer:



5.

5. Create a program on **Thunkable** with a function that calculate the factorial value. The function must receive a parameter and return an output. The program must be able to calculate multiple factorial values at the same time.

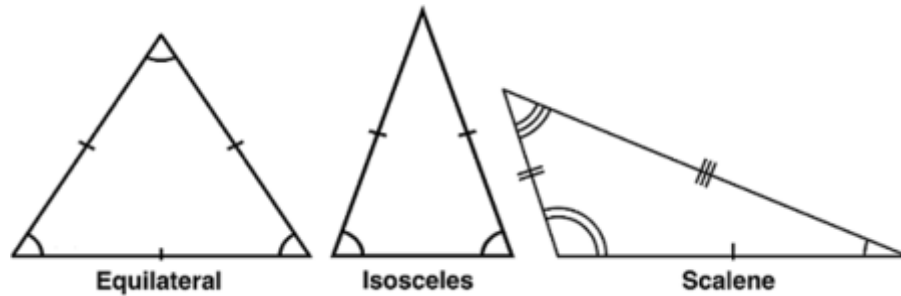
Hint: you need to determine inputs, outputs, and the process of the factorial value.
you need to know which parts should be grouped as a function.

Answer:

The screenshot shows a Thunkable app interface with three input fields containing the numbers 4, 5, and 6. Below these is a blue button labeled "Button". Underneath the button, the calculated factorial values are displayed: 24 for 4, and 120 for 5. The logic blocks are as follows:

- Initialize app variable result to 1**
- Initialize app variable number05_1 to + list - 1, 2, 3**
- when Button4 Click -> do**
 - in list app variable number05_1 set # 1 as Text_Input5's Text x 1**
 - in list app variable number05_1 set # 2 as Text_Input6's Text x 1**
 - in list app variable number05_1 set # 3 as Text_Input7's Text x 1**
 - set Label6's Text to calculateFactorial with: in list app variable number05_1 get # 1**
 - set Label7's Text to calculateFactorial with: in list app variable number05_1 get # 2**
 - set Label8's Text to calculateFactorial with: in list app variable number05_1 get # 3**
- + to calculateFactorial and return with: - variable: x**
 - set app variable result to 1**
 - count with i from 1 to x by 1**
 - do set app variable result to app variable result x i**
 - return app variable result**

6. Create a program on **Thunkable** with a function that checks whether the triangle is an equilateral triangle, isosceles triangle, or scalene triangle. The function must return the word “equilateral triangle”, “isosceles triangle”, or “scalene triangle” to the location that function is called. The program must be able to check multiple triangles at the same time.



Answer:

รูปที่ 1

2 2 2

รูปที่ 2

3 2 2

รูปที่ 3

3 4 5

Button

equilateral triangle

isosceles triangle

scalene triangle

initialize app variable number06_1 to 1
initialize app variable number06_2 to 1
initialize app variable number06_3 to 1

when Button5 Click

do

in list app variable number06_1 set # 1 as Text_Input8 % Text 1

in list app variable number06_1 set # 2 as Text_Input9 % Text 1

in list app variable number06_1 set # 3 as Text_Input10 % Text 1

in list app variable number06_2 set # 1 as Text_Input11 % Text 1

in list app variable number06_2 set # 2 as Text_Input12 % Text 1

in list app variable number06_2 set # 3 as Text_Input13 % Text 1

in list app variable number06_3 set # 1 as Text_Input14 % Text 1

in list app variable number06_3 set # 2 as Text_Input15 % Text 1

in list app variable number06_3 set # 3 as Text_Input16 % Text 1

set Label12 % Text to calculateR with:

x in list app variable number06_1 get # 1

y in list app variable number06_1 get # 2

z in list app variable number06_1 get # 3

set Label13 % Text to calculateR with:

x in list app variable number06_2 get # 1

y in list app variable number06_2 get # 2

z in list app variable number06_2 get # 3

set Label14 % Text to calculateR with:

x in list app variable number06_3 get # 1

y in list app variable number06_3 get # 2

z in list app variable number06_3 get # 3

to calculateR and return with:

if x == y and y == z

do return "equilateral triangle"

else + if x == y or y == z or x == z

do return "isosceles triangle"

else return "scalene triangle"