Lab 04 Boolean Logic and Conditionals

General Information

What you should do

• You should first review Lecture 3 on the topics that we have covered during the lecture and be familiar with the concepts.

• You should try to come up with the code yourself as much as possible. Do not be afraid of making mistakes, since debugging (finding out where your code goes wrong and fixing it) is part of the learning process.

• We do not give out model programs to the exercises. There can be multiple ways to write the code that solves the same problem. It is important that you build up the program logic yourself instead of merely looking at some code that you do not understand. At any time if you are lost or if you have any questions, feel free to ask the instructor, tutor, or teaching assistant and we will be very happy to help you.

Self-Discovery

• Most lab tasks are designed to be relatively simple such that you can take the time to think about the related underlying concepts. Besides, we also encourage you to discover things on your own which may not be specified in the tasks.

Task 1.1 Check if the input is an unsigned integer

In Lab 03 we ask the user to input an unsigned integer in the range of 0-7 (or 0-15). We assumed that the user follows the instruction and enter the proper input. However, sometimes the user may indeed provide other kinds of input which is not meaningful for further processing. For example, a user may input a negative integer, a floating-point number, or even some text. A good programmer should validate the user input and display a warning message if there is something wrong with the user input. The program should only proceed further after the user input has been validated to have the correct form. In this task, you will write a program to check if the input is an unsigned integer and display the result.

Checking if the input is an unsigned integer can be broken down into 2 steps:

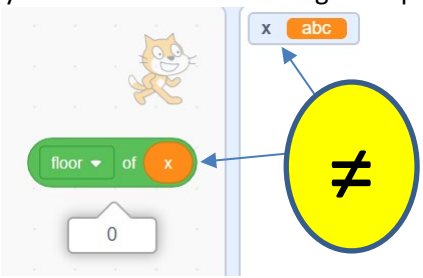
1) Check if the input is an integer
2) Check if the input integer is unsigned

Let us first assume that the input is a number, but this number may not necessarily be an integer as it may be a floating-point number. We need to find a way to detect if a number is integer or not, or alternatively whether the number contains some digits after the decimal point. For this, we will make use of the floor function that was introduced in Lab 03. Recall that the floor function retains the integer part (or removes the decimal part) of a non-negative integer. A more formal way of specifying the floor function is that it is rounding down a number x to the integer such that x-1<floor(x)<=x. With this definition, then you can see that applying the floor function to a negative integer is not simply removing the decimal part, but also subtracting it by 1 afterwards, e.g., floor(-1.5)=-2; floor(-3.9)=-4; floor(-8.2)=-9, etc. Another important observation is that if you take the floor function of an integer, its value would not be changed, e.g., floor(1)=1; floor(0)=0; floor(-3)=-3. On the other hand, you can also observe from earlier examples that applying the floor function to a floating-point number would change its value. As a result, this criterion can be used to detect if a number is an integer or not, i.e., applying the floor function to a number, if its value is not changed, then it is an integer; if its value changes, then it is NOT an integer.

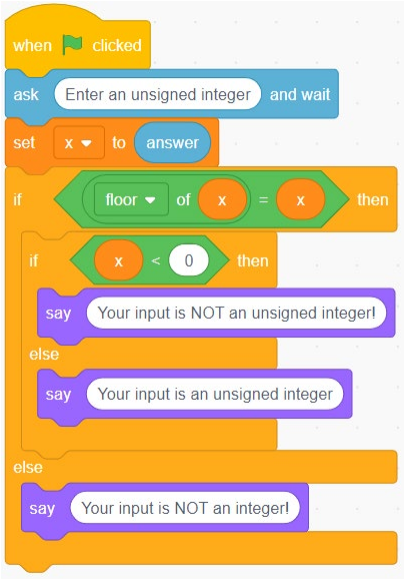
Now create a new project and put down the project name as Lab 04 Task 1.1. Create the variable x and compose the following block:

Run the above program and try with different input numbers. Verify that the program works correctly when your input number is an integer and when your input number is a floating-point number.

How about if the user provides an input that is not even a number? Try running your program by entering some text, e.g., “abc” as the input. You will see that your program still outputs the correct message, i.e., it would say “Your input is NOT an integer”. The reason why the program still works for non-numeric input is that the floor function in Scratch would evaluate anything that is not a number to be 0, which would definitely be different from the original input since it is non-numeric.

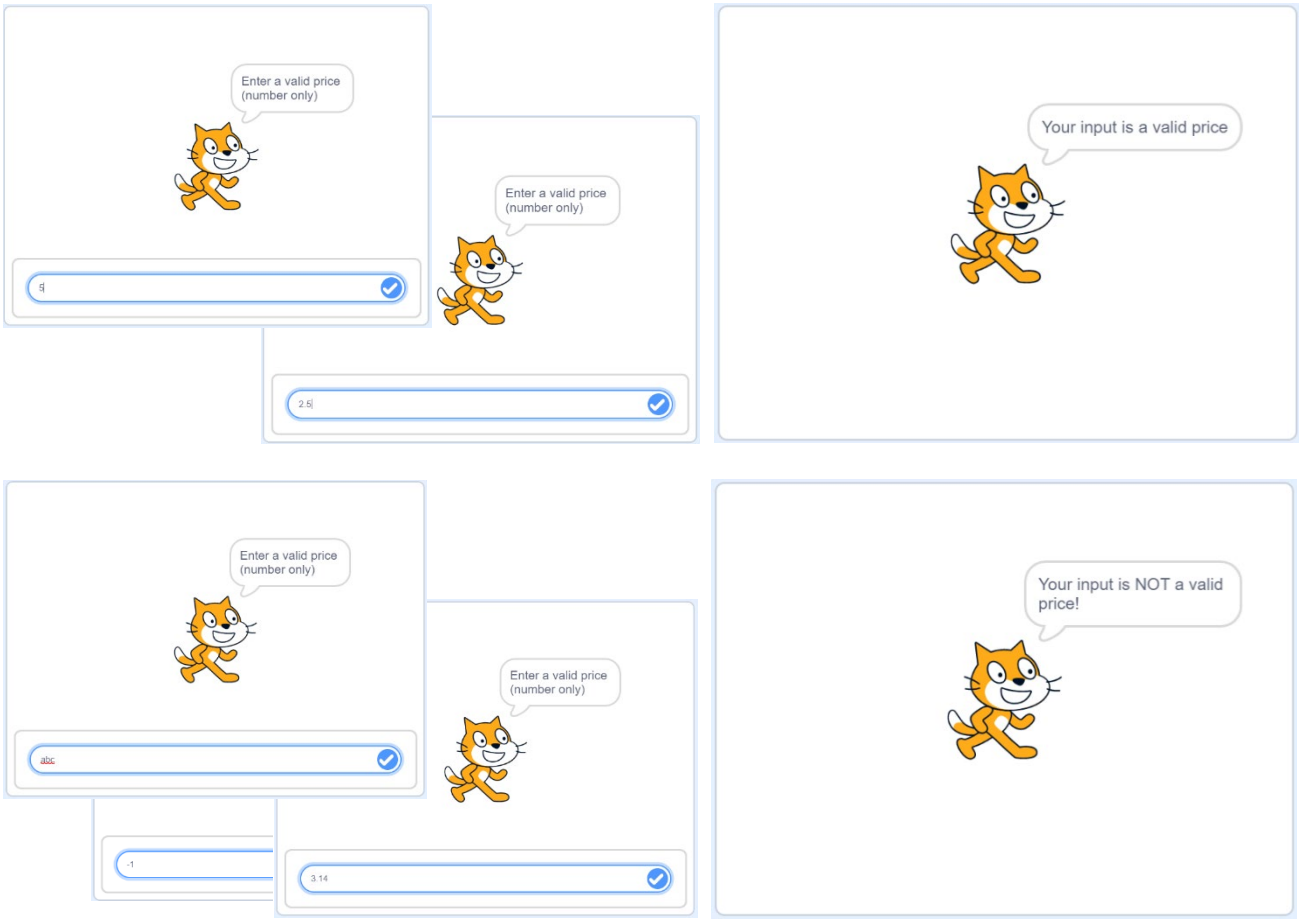


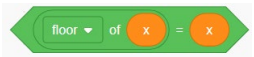
Now let us work on the second step to check if the input integer is unsigned. This part is quite straight forward since it is equivalent to checking whether the input integer is negative. If the input integer is negative, then it is NOT unsigned. Otherwise, the input integer is an unsigned one. Modify the above program by further checking if the input is unsigned after determining that it is an integer. The resulting program is shown below:



Run your program again and try with different input. Verify that the program works correctly when your input is a positive integer, 0, and a negative integer.

Exercise 1.1: Write a program that will ask the user to enter the price of an item as a number. Then the program would check and display the message to indicate if the input is a valid price. A valid price means that it is a non-negative number with at most one decimal place. For example, 0, 5, 2.5, 1.7 are all valid prices and 3.14, -1, "abc", "$5" are not valid prices. Some sample screenshots are shown below:



Hint: You cannot directly use the Boolean block 

as in Task 1.1 to check if the input is an integer or not since a valid price not only includes an integer (but also numbers with one decimal place). If you are thinking of using the Boolean block 

to check if the input contains at most a decimal place or not, this block will only work if x is a number, i.e., it will return true if a number contains at most one decimal place, and false otherwise. However, if x is a text say "abc", this block will still return true because the expression 

will return the value 0 so the left hand side and the right hand side of the above block are both equal to 0. As a result, you should first check if the input is a number or not. To do this, you can use the Boolean block 

. If answer is a number, then obviously the left hand side is equal to the right hand side and this block will return true. If answer is some text (e.g., "abc"), then the left hand side will be equal to 0 but right hand side is still equal to the text so it will return false. If the input is not a number (returned false by the above block), then you should directly display that the input is not a valid price. If the input is indeed a number (returned true by the above block), then you can use the block 

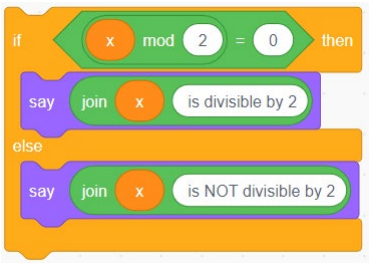
to check if the number contains at most one decimal place. Afterwards, you can further check if the input is non-negative to complete the code.

Task 1.2 Check if an integer is divisible by another integer

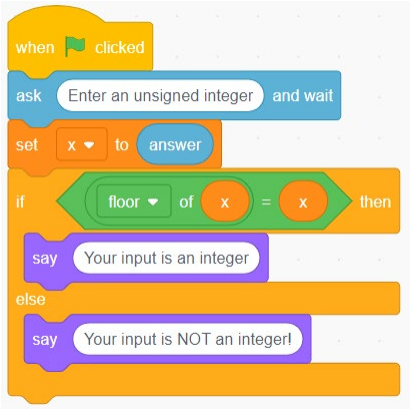
First save a copy of your project and put down the project name as Lab 04 Task 1.2. In this task we would like to check if an integer is divisible by another integer. For example, 6 is divisible by 2 but 5 is not divisible by 2. What is the remainder of the above divisions? The remainder of 6 divided by 2 is 0 and the remainder of 5 divided by 2 is 1. You can observe in general that x is divisible by y if the remainder of the resulting division is 0. Otherwise, if the remainder is non-zero, then x is not divisible by y. Now we need to write the code to check if the remainder is 0 when x is divided by y. Recall in Lab 03 that you have learnt to find the remainder of a division by using the modulus function, i.e., using the block 

.

For now let us assume that the divisor y=2. The following block will check if the input number x is divisible by 2:



Insert the above block to the proper place in the following code (created in Task 1.1) such that the program will first check if x is an integer and then determine if it is divisible by 2 given that x is an integer.



Exercise 1.2: Modify your program so that it satisfies the following additional specifications. You should focus on one specification at a time, test the program each time after you finish with one specification to make sure that it is working correctly before moving to work on the next specification:

a) Replace the divisor 2 by a variable y, whose value should be obtained from the user input. This means that your program should first ask the user to input the dividend and store the answer to the variable x. Then your program should check if x is an integer. If x is an integer, then your program should ask the user to input the divisor and store the answer to the variable y. Then your program should check if y is an integer. If y is an integer, then your program should output whether x is divisible by y.

b) Check if the divisor is 0, then display the message "Divide by 0 error!".

c) Instead of assuming that x is always the dividend and y is always the divisor, your program should just ask the user to input the integers to be stored as x and y. After checking that x and y are both integers, your program should find the bigger number between x and y and store it by the variable m. Similarly, your program should find the smaller number between x and y and store it by the variable n. Afterwards, your program should check and display if m is divisible by n.

Task 1.3 Check for valid floor numbers in a building

A building has its top floor on the 50th floor and has the following floor numbers:
1,2,3,5,6,7,8,9,10,11,12,15,16,17,18,19,20,21,22,23,25,26,27,28,29,30,31,32,33,35,36,37,38,39,50

The other floor numbers within this range (1-50) are not included in this building because 1) the digit 4 is considered as unlucky; and 2) the number 13 is considered as unlucky.

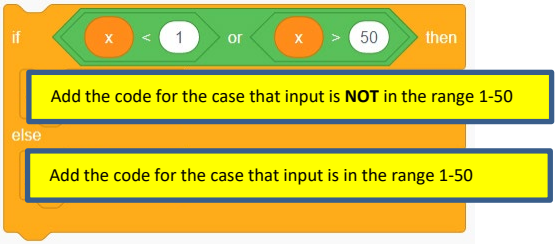
When you try to enter this building, a robot receptionist will greet you and ask you which floor you would like to go. Write a program that simulates the robot receptionist such that

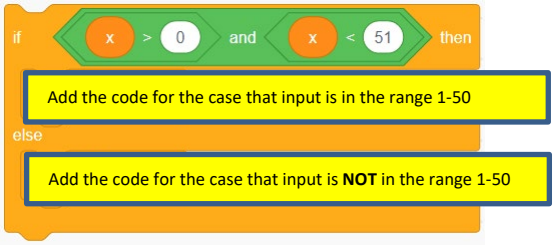
1) It will first ask the user to input the floor number
2) If the floor number is not valid, i.e., if it is not an integer in the above floor number list, then display the message "Your input is NOT a valid floor number!"
3) If the floor number is valid, i.e., if it is one of the integers in the above floor number list, then display the message "Welcome! Please enter!"

Hint:

A) Your program can first check if the input is an integer in the same way as in Task 1.1

B) If the input is an integer, your program can then check if the input is in the range 1-50. There are at least 2 ways you can form the conditional:





C) If the input is in the range 1-50, your program can then check if it is in the above floor number list. From the lecture, we have learnt that we can implement a N-way conditional by nesting N-1 2-way conditionals. The above floor number list has 35 numbers, so theoretically we could form a 35-way conditional by nesting 34 2-way conditionals and checking each time whether the input is equal to one of the numbers in the list. Obviously this would be very tedious and it is easy to make mistakes. As a result, you should identify a smarter way to implement this. You should identify the pattern that describes the numbers in the above list, or alternatively the numbers that are NOT in the above list. We have already mentioned that the numbers in the range 1-50 that are not in the above floor number list either contain a digit 4 or are equal to 13. Now we can implement the code to explore if the input has any of these patterns.

I) To check if the input is equal to 13, you can just use the Boolean block 

II) To check if the input contains a digit 4, we need to examine each digit of the input number. Since it has already been checked that the input is an integer in the range 1-50, the input is at most a 2-digit number. So we need to extract the tens' place and the ones' place from the input and check if each one is equal to 4.

a) Extracting the ones' place

For example, if the input is 5, 12, 34, etc., the corresponding ones' places for these numbers are 5, 2, 4. Can you figure out how to write the code to extract these one's places?

Consider these numbers when divided by 10: 5/10=0…5; 12/10=1…2; 34/10=3…4. You can see that the remainders are exactly the ones' places that we want. As a result, you can use the modulus function to extract the remainder and then compare if it is equal to 4, i.e., forming the Boolean block 

.

b) Extracting the tens' place
If you observe the examples in a), i.e., 5/10=0…5; 12/10=1…2; 34/10=3…4, you can see that the tens' place of a 2-digit number is in fact the quotient of that number when divided by 10. In other words, we can obtain the ten's place of the input which is a 2-digit number by first dividing by 10 and keeping only the integer part. This can be implemented using the floor function, i.e. forming the block 

.

III) You can make use of the Boolean blocks derived in I), IIa), IIb) and check it one by one. If a Boolean block returns true, then you know that the input is not a valid floor number. Alternatively, you can use Boolean operators to join these 3 Boolean blocks and form a compound Boolean block so that you can use a single 2-way conditional to check for this. Should you use the OR operator or the AND operator to join these 3 Boolean blocks?

Exercise 1.3: Modify your program so that it satisfies the following additional specifications:

4) After verifying that the input is a valid floor number and saying "Welcome! Please enter!" for 2 seconds, the robot receptionist will give you further directions:
   i) If your floor number is among 1,2,3,5,6,7,8,9,10,11,12,15, then the robot receptionist will say "Please proceed to take the lift on the left."
   ii) If your floor number is among 16,17,18,19,20,21,22,23,25,26,27,28, then the robot receptionist will say "Please proceed to take the lift on the middle."
   iii) If your floor number is among ,29,30,31,32,33,35,36,37,38,39,50, then the robot receptionist will say "Please proceed to take the lift on the right."

Task 1.4 Recommend a travel option

Suppose that you need to go from Location A to Location B, and you have the following travel options with the corresponding transportation costs:

	Travel Option	**Transportation Cost ($)**
1	On foot	0
2	By tram	2.6
3	By subway	10.7
4	By bus	11.1
5	By taxi	130

Write a program with the following specifications:

a) When the program starts, it will ask the user how much money the user has
b) The program will then check if the user input is a valid amount of money. A valid amount of money means that it is a non-negative number and contains at most 1 decimal place. It will display a message "Your input is NOT a valid amount of money" if it is the case and the program ends.
c) If the user input is indeed a valid amount of money, then the program would recommend the most expensive travel option that the user can afford, by displaying a message "You can go from Location A to Location B [travel option], and it will cost you [transportation cost]".

Example 1:

User input: 5.65
Message displayed: Your input is NOT a valid amount of money

Example 2:

User input: 5
Message displayed: You can go from Location A to Location B by tram, and it will cost you $2.6

Example 3:

User input: 10.8
Message displayed: You can go from Location A to Location B by subway, and it will cost you $10.7

Test your program by using different input values. Make sure you include enough variations of the input such that each travel option appears at least once during your testing.

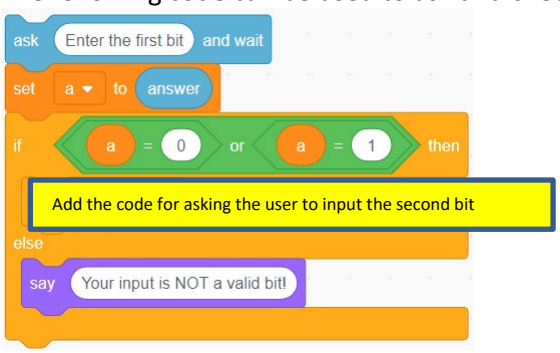
Hint:

A) Refer to Exercise 1.1 for checking if the input is a valid amount of money
B) Form a 5-way conditional by nesting 4 2-way conditionals

Task 1.5 Boolean Logic

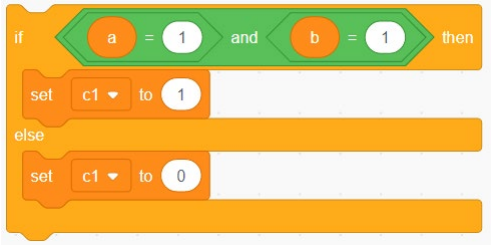
In this task you will write a program that will first ask the user to input one bit a, and then input another bit b. The program will check if each user input is indeed a bit, i.e., 0 or 1. After verifying that the two user inputs are valid bits, then the program will compute and display c = a AND b. We will use 2 different methods to calculate c: 1) using conditional; 2) deriving an equation.

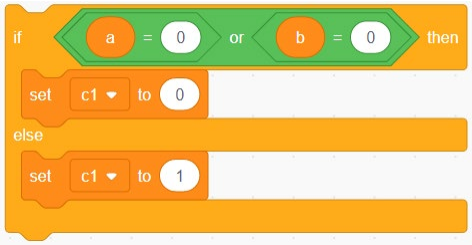
A) Getting and checking user input
The following code can be used to ask and check if the user input a valid bit:



Now starting from the above program, you add the code to ask the user to input the second bit, store the answer as b and check if it is a valid bit in a similar way as the first bit.

B) Using conditional to determine the output of the Boolean logic
Suppose that we use the variable $c1$ to store the result of a AND b in this part. You have learnt that the output of the AND gate is 1 when all inputs are 0. Alternatively, as long as any input bit is 0, then the output of the AND gate is 0. As a result, $c1$ can be determined by the code either on the left hand side or on the right hand side (so you can choose either one to implement):





C) Dering an equation to determine the output of the Boolean logic
Suppose that we use the variable $c2$ to store the result of a AND b in this part. Instead of using conditional, we will derive an equation for a AND b. The truth table of a AND b is given below:

a	b	$c2$ = a AND b
0	0	0
0	1	0
1	0	0
1	1	1

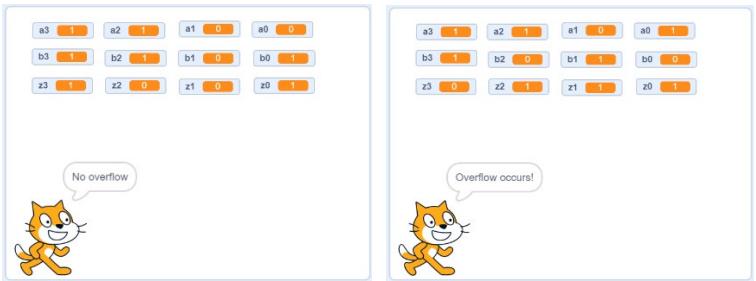
It should be obvious from the above table that $c2$ can be obtained by multiplying a with b. Thus you can use the following code to determine $c2$:

Complete the code and run your program by entering the input bits from each row of the above truth table. In each case verify that the values of $c1$ and $c2$ agree with the above truth table.

Exercise 1.5: Follow the same procedure as the above for other logic gates NAND, XOR, XNOR, OR, NOR. In each case, implement it using conditional as in B) and also try to derive the equation as in C).
Hint:

i) For deriving the equation for NAND, examine the truth table and identify the relationship between the values of AND and NAND in each row.
ii) For deriving the equation for XOR, refer to Lab 03 Task 1.2B 2).
iii) For deriving the equation for XNOR, start from XOR.
iv) For deriving the equation for OR, if you are not able to determine how to relate the output with the input, then try considering the sum $a + b + ab$, i.e., adding the 3 columns of the truth table on top of this page. Then try to see if you can turn the resulting sum to the result of the OR operator by using one operation that you have been using for Lab 03 and this lab. Note that this is only one of the many possible ways to derive an equation. You may come up with another equation which is perfectly fine as long as it is correct.

Exercise 1.6: Extend your program from Lab 03 Task 1.3 by adding the code to check and display if the addition of the two 4-bit binary numbers result in overflow.



Task 2 Complete the assessment from the Canvas course page

You should complete the Lab 04 Assessment from the Canvas course page before the posted deadline.

Task 3 Challenge your classmates

You can first reflect on what you have learnt in this lab, and then come up with problems to challenge your classmates. You can post your problem on the Canvas course page, under this Discussion page. One should be able to solve your problem by using what he/she learns in Lab 04. You will not get extra marks by posting a challenging problem or solving a challenging problem posted by another student, but you will earn your fame so that you can impress the course leader, the lab tutors, and your classmates.
