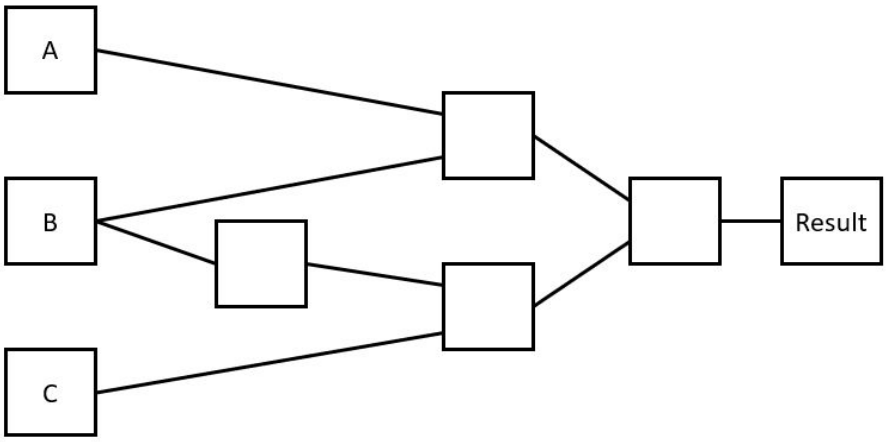
These problems were generated by TAs and instructors in previous semesters. They may or may not match the actual difficulty of problems on Quiz2.

Circuits and Gates

1. Given the following boolean expression, fill out a truth table that shows all the possible results of the expression, then label the gates on the circuit below with AND/OR/etc. so that it produces the same results.

(A or B) and ((not B) xor C)

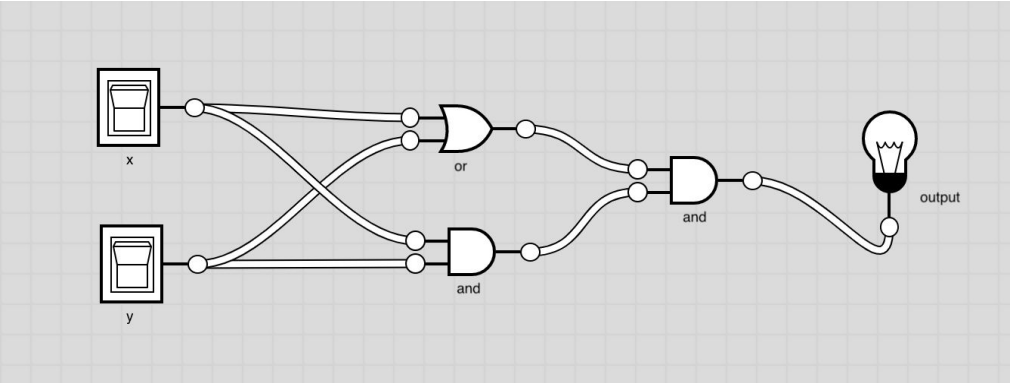
Circuit:



2. Recall that in lecture we built a simple addition machine called a Full Adder. Clearly name and describe the purpose of the input(s) and output(s) of this machine.

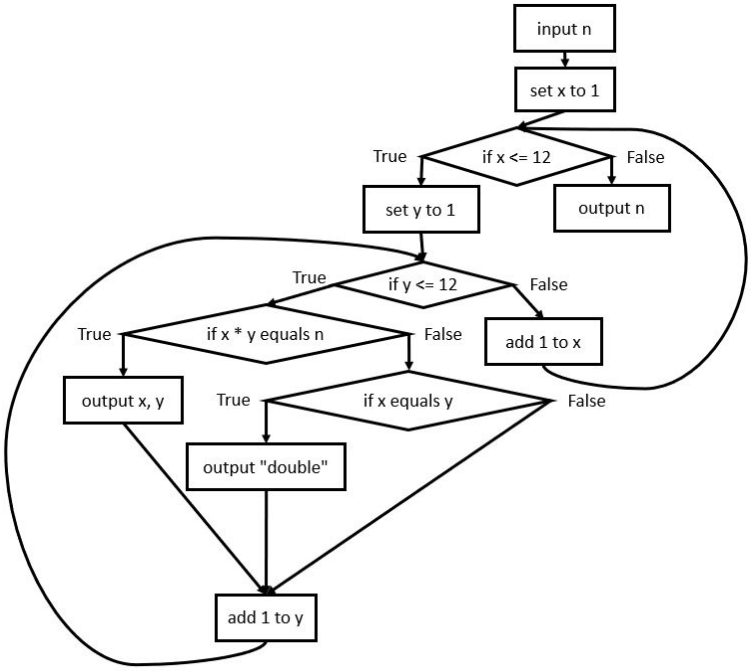
3. What is the main difference between a half adder and a full adder?

4. What boolean operation does the following logic circuit behave like?

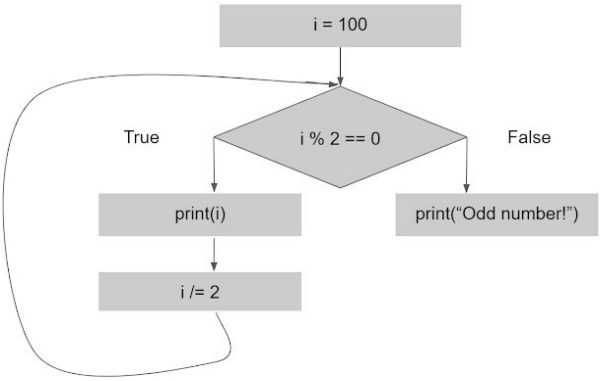


While Loops

1. Write a function `cw1(n)` that is algorithmically identical to the control flow chart shown below. The function should take an integer n as a parameter and **print** output as specified, returning nothing.



2. Write the while loop that corresponds with this flow chart.



3. Use while loop to write the function `hasConsecutiveDigits(n)` that takes in a possibly-negative int value n and returns True if that number contains two consecutive digits that are the same, and False otherwise.

4. Write the function `isPowerOfFour(n)` that takes in a number n and returns True if n is a power of 4, and returns False otherwise.

For Loops

1. Explain when you would use a while loop versus a for loop. Can you always convert a for loop to a while loop? Can you always convert a while loop to a for loop?

2. Write a function `numberOfFactors(n)` which takes in a positive integer and returns the number of factors it has.

3. Using a for loop, write the function `fizzBuzz(n)` that prints every number from 0 to n-1 inclusive. If the number is divisible by 3, print “fizz” instead of the number. If the number is divisible by 5, print “Buzz” instead of the number. If divisible by both 3 and 5, print “fizzBuzz” instead of the number.

4. Using a for loop, write the function `sumAllEven(n)` that finds the sum of all even numbers less than or equal to n.

Strings

1. Read through the following block of code, and write what it will output..

```
s = "Computer Science"
t = "GO-1-TEN"

print("A:", s[4])
print("B:", t[len(t)-2])
print("C:", s[6:12])

print("D:", s > t)
print("E:", s.find("e"))
print("F:", t.lower())

for i in range(2, 10, 4):
    print(s[i] + t[i])
```

2. Write a function `whileSmile(s)` that takes a string as input and uses a **while loop** to count the number of times the two-character string ":)" occurs in s. You should return the count. For example: `whileSmile("Hello :):):)")` should return 3. **Do not use the built-in function s.count().**

3. Write a function `reverseString(s)` that returns a reversed version of the string s.

Lists

1. Write the function `onlyNegative(L)` which takes a list of numbers, `L`, and returns a **new list** that contains only the negative numbers in `L`, in the order they originally appeared. Your function should **not** destructively modify the original list `L`.

For example, `onlyNegative([-2, 1, 2, -1, 0, -3, 3])` would return the list `[-2, -1, -3]`.

2. Write a function commonElement that takes in two lists and returns True if they share a common element (ie. there is an element that occurs in both lists).

3. Given a word search puzzle (format is a 2D list of strings), check to see if a given word is in the board. Words can be left->right or up->down, but not right->left, down->up, or diagonal. For example,

```
puzzle = [ [ 'a', 'q', 'r', 't' ],
           [ 'd', 'o', 'g', 'a' ],
           [ 'w', 'm', 'z', 'c' ] ]

assert(wordSearch(puzzle, "dog") == True)
assert(wordSearch(puzzle, "cat") == False)
```