# Final Exam
# Algorithms and Programming for High Schoolers (AddisCoder)

| Problem | Score | Max Score |
|---|---|---|
| 1 | | 23 |
| 2 | | 11 |
| 3 | | 11 |
| 4 | | 11 |
| 5 | | 11 |
| 6 | | 11 |
| 7 | | 11 |
| 8 | | 11 |
| Total | | 100 |

Name:

Contact (e-mail address or phone number):

Name:

**Question 1:** Imagine evaluating the following expressions in order in the Python interpeter. For each expression in red, write down what the expression would evaluate to in the space below. If the expression would cause an error in Python, then write Error. Each answer is worth 1 point.

```
>>> x = 5

>>> y = 7

>>> z = 2

>>> y / x - z


>>> (y / x) - z


>>> y / (x - z)


>>> y % (x - z)


>>> y % x


>>> [[[]]][0]


>>> len([[1,[2,[3]]],[4],[5]])


>>> len([[1,[2,[3]]],[4],[5]][0])


>>> L = [[1,[2,[3]]],[4],[5]]
```

Name:

```
>>> L + L[0]


>>> L + L[0][0]


>>> 'Ethiopia'[3]


>>> 'Ethiopia'[3][0][0][:]


>>> 'abcd'[:2]


>>> 'abcd'[2:]


>>> 'abcd'[1:2]


>>> int('2')


>>> w = n + 1
>>> w


>>> y = []
>>> w = 10
>>> while w > 0:
>>>    y += [[w]]
```

Name:

```
>>>    w /= 2
>>> y


>>> def fibonacci(n):
>>>    if n < 2:  return 1
>>>    return fibonacci(n-1) + n - 2
>>> fibonacci(-2)


>>> fibonacci(4)


>>> fibonacci(5)


>>> [] != [[]]


>>> x = 7
>>> x %= 2
>>> x
```

Name:

**Question 2:** Write a function `countZeroes`(n) which takes as input a positive `int` n and outputs the number of zeroes in n. For example, `countZeroes`(10) is 1, `countZeroes`(50803) is 2, and `countZeroes`(547) is 0. What is the running time of your algorithm in terms of the number of digits $D$ in $n$?

Name:

**Question 3:** You are given a `list` of pairs `L = [[`$x_0$`,`$y_0$`], [`$x_1,y_1$`], ...,` `[`$x_{n-1}$`,`$y_{n-1}$`]]` such that $y_i = x_{i+1}$ for all $0 \leq i \leq n-2$. When you combine adjacent pairs $[x_i, y_i]$ and $[x_{i+1}, y_{i+1}]$ (recall $y_i = x_{i+1}$), the new pair $[x_i, y_{i+1}]$ takes their place in the `list`. Combining this pair has cost $x_i \times y_i \times y_{i+1}$. You need to keep combining adjacent pairs until you're finally left with the single pair $[x_0, y_{n-1}]$, but you can choose the order in which you combine adjacent pairs. Write a function `bestCost(L)` which calculates the minimum cost of how to do this. What is your running time? (Justify your answer.)

For example, consider the input `L = [[1,5],[5,3],[3,7]]`. If you first combine `[1,5]` and `[5,3]`, the cost is $1 \times 5 \times 3 = 15$. Then you are left with the `list [[1,3],[3,7]]`, and combining these two has cost $1 \times 3 \times 7 = 21$. Thus the total cost is $15 + 21 = 36$. The other option is to first combine `[5,3]` and `[3,7]`, for a cost of $5 \times 3 \times 7 = 105$, producing the new pair `[5,7]`. The list is then `[[1,5],[5,7]]`, and combining these has cost $1 \times 5 \times 7 = 35$, for a total cost of $105 + 35 = 140$. Thus, the first option was better, and `bestCost([[1,5],[5,3],[3,7]])` should return 36.

Name:

Name:

**Question 4:** You are given a chessboard which has $n$ rows and $m$ columns. The bottom-left square is $(0,0)$, and the top-right is $(n-1,m-1)$. You want to get your piece from the bottom-left to the top-right. If your piece is at $(x,y)$, the next step it can either move to $(x+1,y)$, $(x+1,y+1)$, or $(x,y+1)$, as long as the piece stays on the board. There is one catch though. Squares on your chessboard are either green or red, and on odd moves (your first, third, fifth, etc. moves) you can only move to red squares, and on even moves you can only move to green squares. If ever you can't make a move because all squares next to you are the wrong color, then your piece dies.

Write a function `isPossible(n,m,colors)` which returns `True` if it is possible to get from the bottom-left to the top-right corner without dying and `False` otherwise. Also, before writing the code, describe in words how your solution works: your description should not take more than a couple sentences. `colors` is a `list` of $n$ strings each of length $m$. `colors[i][j]` is `'R'` if square $(i,j)$ is red, and otherwise is `'G'`. For example, `isPossible(3,3,['GGG','RRG','GGG'])` gives `True`. The board is

| | | |
|---|---|---|
| G | G | G |
| R | R | G |
| G | G | G |

Name:

Name:

**Question 5:** Describe an algorithm that does the following. You are given an integer $n \geq 2$ and must return the index $i$ of the first Fibonacci number which is larger than $n$. Recall the Fibonacci numbers are $F_0, F_1, F_2, F_3, \ldots = 1, 1, 2, 3, \ldots$ (each number is the sum of the previous two). So, if $n = 7$, then the answer should be 5: the first Fibonacci number larger than 7 is $F_5 = 8$. If $n = 2$, then the answer should be 3, since the third Fibonacci number $F_3 = 3$ is the first Fibonacci number to be larger than 2.

You don't need to implement your solution, but you should describe how you would do it and also explain the running time assuming that all arithmetic operations take $O(1)$ time.

Name:

**Question 6:** You are given a `list L` of `int`s that are all bigger than 1. Write a function `findPair(L)` that returns a list `[a,b]` such that $a^2 = b$ and `a`, `b` are both in `L`. If no such pair exists, return `[]`. For example, `findPair([9,5,2,7,3])` should return `[3,9]`. `findPair([5,2,25,4])` can either return `[2,4]` or `[5,25]`. `findPair([9,12,14])` should return `[]`.

Name:

**Question 7:** Given a directed graph where each edge has a length, describe an algorithm that takes as input two vertices $u, v$ and an integer $k \geq 0$ and outputs the length of the shortest path from $u$ to $v$ which takes *exactly* $k$ steps. The path is allowed to visit vertices multiple times (for example, the path $1 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 7$ is a valid path from 1 to 7 of length 4, even though it visits vertex 3 twice). Furthermore, on odd moves (the first, third, fifth, etc. moves), you must take the edge out of your current location which is the longest (assume that no two edges have the same length). What is the running time of your algorithm? You do not have to write the code for it.

Name:

**Question 8:** We've discussed making change using the least number of coins possible. What if we want to count how many different ways there are of making change? Furthermore, we want to count the number of different ways of making change when we're only allowed to use an *even number of each coin type*. For example, if the coins we have available are `[1,5,10,25]` cents and we want to make change for 12 cents, there are 2 ways: (1) give twelve 1-cent pieces, and (2) give two 5-cent pieces and two 1-cent pieces. The other ways would involve giving an odd number of some coin, so we can't do it.

Write a function `change(L,n)` which outputs the number of ways to make change for n cents when the coin denominations available are those in L. For example, `change([1,5,10,25], 12)` should return 2. What is the running time of your solution?

Name: