Iverson Computing Competition - Solutions
2022, Monday, June 6

name(s) _____

_____

_____

school _____

city _____

grade(s) _____

supervising teacher _____

supervising teacher email _____

individual or group? _____

Write your responses legibly.
Read the instructions sheet carefully (sent separately).

question	- - - -	marks	your score
1	spice thief	15	
2	birthday candles	15	
3	powtwo game	15	
4	a binary sequence	15	
total	- - - -	60	

Prepared by Ian DeHaan, Zac Friggstad, Kaaden RumanCam, and Noah Weninger.

Exam Instructions

This exam is to be written without the assistance of electronic tools apart from a simple, nonprogrammable calculator. In particular, you may not use a search engine nor may you implement or run any programs. If we were in person then you would write this exam with only a calculator and a pencil or pen. Do the same from wherever you are writing this exam. Your mobile device should be stowed away for the exam.

Some questions are easy, some are challenging. Read each question carefully so you fully understand the task. Some questions ask for pseudocode. You can choose to write either code in your preferred programming language or you can use high-level pseudocode. If you use pseudocode, a programmer should be able to easily convert your pseudocode into an implementation in a programming language.

In this exam, we are more interested in algorithmic thinking than programming language prowess. Syntax errors will be ignored as long as it is clear what you were trying to accomplish.

Be concise, but still complete, when providing answers.

If you need more space, you may use the blank pages at the end of this exam. Just clearly indicate when you have done so by writing where we can find your solution (eg. page number).

question 1: spice thief

A thief plans to break into a spice store and steal spices to sell on the black market. After careful observation, the thief has determined how many grams of each spice are in the store and their selling prices (per gram). The thief has devised a plan to maximize their profit based on the limited capacity of their magical bag which stops spices from mixing together.

As an example, suppose the thief's bag can hold 10 grams of spices, and the following four spices are in the store:

Number of Grams	5	7	5	1
Price per Gram	3	4	2	2

Then the thief should take 3 grams of the first spice and 7 grams of the second spice for a total profit of (**corrected typo)** $3 \cdot 3 + 7 \cdot 4 = 37$. You can check that any other way of gathering at most 10 grams of spices will not yield a profit greater than 37.

a) [6 marks] Find the optimal amount of profit for each of the following cases.

i) The capacity of the bag is 5 grams, and the following spices are in the store:

Number of Grams	3	3
Price per Gram	3	4

Solution: Take 3 grams of the second spice and 2 grams of the first spice. Profit = $2 \cdot 3 + 3 \cdot 4 = 18$.

ii) The capacity of the bag is 10 grams, and the following spices are in the store:

Number of Grams	3	3	1	2
Price per Gram	3	4	2	2

Solution: You can fit everything in the bag. Profit $= 3 \cdot 3 + 3 \cdot 4 + 1 \cdot 2 + 2 \cdot 2 = 27$.

iii) The capacity of the bag is 5 grams, and the following spices are in the store:

Number of Grams	3	3	1	2
Price per Gram	1	2	4	3

Solution: Take all of spice 3, all of spice 4, and then 2 grams of spice 2. Profit = $2 \cdot 2 + 1 \cdot 4 + 2 \cdot 3 = 14$.

b) [4 marks] Describe a strategy for selecting spices to steal that will always yield an optimal profit.

Solution: Take as much of the spice with greatest price per gram as you can fit in the bag. If there is leftover space in the bag, repeat with the remaining spices.

In other words, always take 1 gram of the highest price per gram spice that remains until the bag is full or until you take all spices.

c) [5 marks] Give pseudocode for a function `steal(C, N, A, P)`. Here, `C` denotes the maximum capacity of the bag in grams. `A` and `P` are lists of size `N` containing the number of grams and price per gram of each spice, respectively. The function should return a single integer denoting the optimal profit.
Example: The sample input at the start of the question would look like `steal(10, 4, [5, 7, 5, 1], [3, 4, 2, 2])`, and the function would return `43`.

Solution: See the included code `q1.py`

question 2: birthday candles

Humans of planet Earth tend to count using base 10 since they have 10 fingers. This means incrementing through all the numbers {`0, 1, 2, 3, 4, 5, 6, 7, 8, 9`} in one position before incrementing the digit in the next position. Consequently, a human with the three birthday candles `1 1 2` on their cake will know that they are turning $1 \cdot 10^2 + 1 \cdot 10^1 + 2 \cdot 10^0 = 100 + 10 + 2 = 112$ years old. Ivs of planet Erson count in a similar manner except that they don't necessarily have 10 fingers. In fact, the number of fingers that each Iv has can be any value at least 2! It is very important that the birthday candles of an Iv's birthday cake are correct for the number of fingers that they have or else they will not be able to tell how old they are. Can you help Human space emissaries carry out important birthday celebrations?

a) [5 marks] Determine the *candle representation* for each Iv in the table below. That is, what digits would be used to display the corresponding Iv's age if it were displayed using the number system corresponding to the number of fingers that Iv has. For digits with value > 9 please use letters {`A, B, C, ..., Z`} where `A` represents 10, `B` represents 11, and so on.

Example: for the first row we have that the Iv with 11 fingers is turning $43 = 33+10 = 3 \cdot 11^1 + 10 \cdot 11^0$ years old. The 11 digits used to represent the values 0 through 10 are `0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A`. Thus, the two candles that would be used are `3` and `A`.

turning age	# of fingers	candle representation
43	11	3A
14	3	112
68	35	1X
11	2	1011
1055	16	41F
4694	20	BEE

b) [7 marks] Write a function `candles(age, fingers)` that returns a string denoting the candles for a birthday cake for an Iv turning `age` years old and that has `fingers` number of fingers. You can assume that `fingers` is an integer between 2 and 36.

Solution: See the included code `q2.py`

c) [3 marks] You must plan a surprise party for an Iv turning 165. However, you do not know how many fingers this Iv has. Thankfully you do know that the rightmost candle on their cake will be a 0. List the possible numbers of fingers that this Iv could have, even those that are greater than 36.

Example: 165 itself should be among your list of a possible number of fingers for the following reason: if the Iv has 165 fingers then the two candles would simply be 1 0 since $165 = 1 \cdot 165^1 + 0 \cdot 165^0$.

Solution: There is a 0 at the end of the candle representation if and only if the number of fingers is a divisor of 165 and is ≥ 2 (since Ivs do not have 1 finger in total). An easy way to find all divisors of 165 is to first factor it into its primes:

$$165 = 3 \cdot 5 \cdot 11$$

and then try all ways of using at least 1 prime to form a divisor:

- Just 1 prime: $3, 5, 11$
- 2 primes: $15, 33, 55$
- All three: 165

So there are 7 possible numbers of fingers an Iv could have such that having that many fingers would cause the candle representation of 165 have the rightmost digit be 0.

question 3: powtwo game

Alice and Bob are playing a game. This game starts with some amount n of stones in a pile. On any player's turn, they must remove some power of two $(1, 2, 4, 8, \ldots)$ stones from the pile. More precisely, if the pile currently contains m stones then the next player to play chooses to remove 2^i stones for some integer $i \geq 0$ provided $2^i \leq m$. If the next player to play cannot make a move, i.e. the pile is empty, then they lose and the other player wins. Alice plays first.

a) [3 marks] If the pile starts with 0 stones, then Bob wins because Alice cannot make a move. If the pile starts with 1 stone, then Alice has a winning strategy because she can take $2^0 = 1$ stone from the pile which leaves no move that Bob can make on his turn. Similarly, if the pile starts with 2 stones then Alice again has a winning strategy because she can take $2^1 = 2$ stones. Fill in the rest of the following table with A or B indicating who would win if both Alice and Bob play perfectly.

n	0	1	2	3	4	5	6	7	8	9	10	11
Winner	B	A	A	B	A	A	B	A	A	B	A	A

b) [5 marks] Give pseudocode for the function `winner(n)` which prints who will win (`A` or `B`) given that there are initially `n` stones in the pile and both Alice and Bob play perfectly. Any pseudocode that correctly answers the problem will be given partial credit. For full credit, your algorithm should return an answer in a fraction of a second if implemented and run on a desktop computer even if n is very large (i.e. around 10^{15}).

Solution: See the code `q3partb.py` for 2 solutions, one worth partial credit and one worth full credit.

c) [2 marks] Briefly explain why your algorithm will work.

Solution: Alice will win if and only if the initial number of stones is not a multiple of 3. To see this, note that if the initial number of stones is not a multiple of 3 then Alice can play 1 or 2 stones to reduce it to a multiple of 3. Then no matter what Bob plays, the resulting pile will not have the number of stones reduced to a multiple of 3 since 2^i is not a multiple of 3 for any integer $i \geq 0$. So when the pile comes back to Alice, she can repeat the strategy of ensuring she always plays on a pile where the number of stones is not a multiple of 3. Therefore, Alice will never be given a pile with 0 stones (i.e. she will not lose).

If the initial number of stones is not a multiple of 3, then Bob will win by the same argument except noticing Alice has no choice but to give Bob a number of stones that is not a multiple of 3 and then Bob can play a 1 or 2 to ensure it becomes a multiple of 3 again.

d) [5 marks] While Alice and Bob enjoyed playing the powtwo game, they now find it boring because they have figured out the optimal strategy. To fix this, Alice has come up with a brilliant new game that she calls the oddpow game. Before the start of the game, Alice and Bob will agree on an odd number $(1, 3, 5, 7, \ldots)$ which we denote by k. After this, the game is played exactly the same as the powtwo game, except that on a player's turn they must remove some power of k $(k^0, k^1, k^2, \ldots)$ stones from the pile.

Example: Suppose Alice and Bob agree on $k = 3$. If the pile starts with 0 stones, Bob wins just like in the powtwo game. If the pile starts with 1 stone, then Alice can take this stone and win the game. If the pile starts with 2 stones, then the only move Alice can make is to remove 1 stone. Bob will then remove 1 stone and win the game, so Bob wins if the pile starts with 2 stones.

Your Task: Essentially, repeat part b) with this new game. Implement the function `winner(n, k)` which prints who will win (`A` or `B`) given that there are initially `n` stones in the pile, Alice and Bob have agreed on using powers of `k` (`k` is guaranteed to be odd), and both Alice and Bob play perfectly. However, your solution must not use loops or recursion.

Also explain briefly why your pseudocode is correct.

Solution: See `q3partd.py`. The explanation is simpler than before. Alice wins if and only if the pile she is initially given has an odd number of stones. This is because no matter how anyone plays, the parity of the pile (even or odd) changes since k^i is an odd number for any integer $i \geq 0$ and any odd integer k.

question 4: a binary sequence

Consider the following method to construct an infinite sequence of zeros and ones (bits). We start with our sequence containing a single `0`. Then, we repeat the following: make a copy of the entire sequence, invert the copy (i.e., swap `0` and `1`), and append the inverted copy to the end of the sequence.

For example, at the first step, we copy the single `0`, invert it to a `1`, and add it to the end to get `01`. Repeating this process, the next few steps yield:

```
0110
01101001
0110100110010110
...
```

Evidently, this process can repeat forever to produce an infinitely long sequence. We can number the positions so that we can refer to specific bits in the sequence. Let a_i be the i^{th} bit in the sequence (using 0-based indexing). For example, $a_0 = 0$, $a_1 = 1$, $a_2 = 1$, $a_3 = 0$, $a_4 = 1$, and so on.

a) [2 marks] The first 16 bits of the sequence were written above, namely `0110100110010110`. Write the next 16 bits.

Solution: `1001011001101001`

b) [10 marks] Write a function $f(n)$ which returns the value a_n for any integer $n \geq 0$. Any function that correctly computes a_n will receive partial marks. For full marks, your function should run in a fraction of a second if implemented and run on a modern desktop machine even if n is around 10^{15}.

Solution: See `q4.py` for two solutions, one worth partial credit and one worth full credit.

The full credit one just computes the number of 1 bits in the binary representation of n and returns 0 if this is even, or 1 if this is odd. We didn't ask you to explain *why* this works, but here is a very quick argument. Suppose it works for the first 2^i numbers $a_0, a_1, \ldots, a_{2^i-1}$. The next 2^i numbers are obtained by flipping the bits of the previous 2^i numbers (like you did in the first part). But we can write any $2^i \leq n < 2 \cdot 2^i$ as $2^i + n'$ where $n' < 2^i$. Since $a_{n'}$ is the parity of the number of 1-bits in the binary expansion of n' and since the binary expansion of n has one more 1-bit than n', we see that a_n is 0 or 1 according to the number of 1-bits in its binary expansion as well.

c) [3 marks] What is a_{2022}? Explain why it is possible to easily find the value of a_n by hand, even when n is large. Only answers with explanations will receive marks.

Solution: Write 2022 in binary: `11111100110`

We see the number of 1s in this binary expansion is 8, so a_{2022} is 0. In general, we can compute a_n easily by writing n in binary and counting the number of 1s. We can write n in binary easily by hand by repeatedly dividing it by 2 and using the remainders (0 or 1) to build the binary representation.