ProCo 2019 Round 2

Avengers: Endgame

This year's Round 2 packet will include problems in a Capture the Flag (CTF) style. These problems will be covered in a variety of topics which include **cryptography, optimization, script writing, assembly language,** and **miscellaneous**.

The Avengers find themselves reeling from not only the global crisis at hand but also from their fallen friends. In disarray, they have decided to band together to fight Thanos in one final battle after finding him on a nearby planet: The Garden. In their journey to find Thanos, they encounter a large number of obstacles along their way. Thanos has not made it easy to find him and has laid numerous traps along the travel. With Tony Stark incapacitated, it is up to the genius of your team to solve these puzzles.

Ciphers

Ali(p)en Attack (100)

The Avengers run into the worst of the Ravagers: The Brown Pigs. With Captain Marvel scouting far ahead, they are ensnared into the trap of the Pigs' mother ship. Desperate for help, they call upon you to help them to break the trap, but the trap has a password:

Substitute the Wire (200)

Days later, the Avengers run into the first of several traps Thanos has set along the way. They become trapped within the Chitauri's small legion consisting of two space dinosaurs and several ships with their light speed drive shut down. Fortunately, they decide to come up with a makeshift EMP device that their ship is shielded from, but they forgot how to work it. Tony left behind a manual but it seems to be encrypted with his favorite password.

https://drive.google.com/file/d/1qOP7Ot0i4kCmjZJFJtF2T2D0RVtnzhth/view?usp=sharing

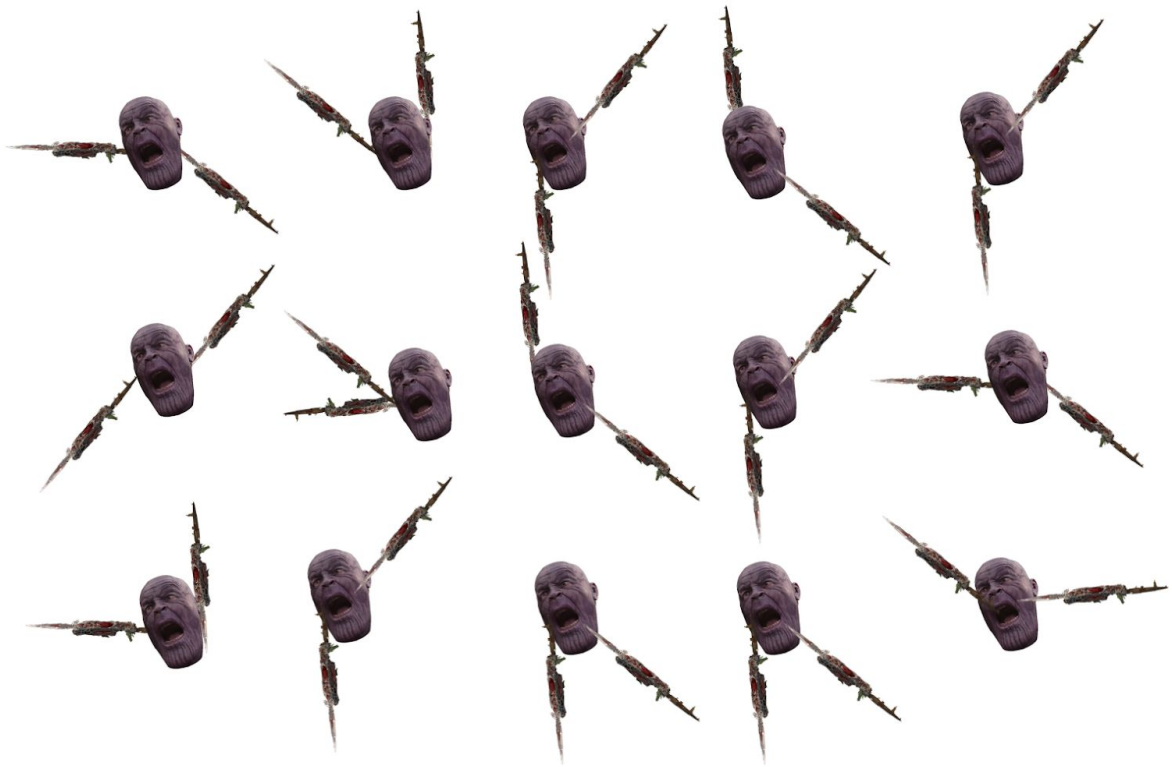
Custom Language (300)

The Avengers reach their next destination: The Galactic Border enforced and protected by the legendary Transgalactic Security Agents. To proceed to the designated checkpoint where the Avengers can get their much needed supplies, they must undergo screening. On the way through the screeners, Rocket is detained due to some unfortunate circumstances and you are brought in to help negotiate his release. After a desperate plea and a long rambling discussion about the fate of the universe (blah blah blah), the TSA decide to let the entire team go if you could help them crack a simple code. The code is a little weird as it uses a language you've never seen before, but it seems like the alphabet is: poiuytrewqmnbvcxzlkjhgfdsa0987654321.

The code is: 4915b2hqs61e9zpp8ezb2cpg3v

セマフォル (300)

During Nebula’s time at Earth, she has had the opportunity to visit many different countries, one of which was Japan. She brought along a custom entertainment system for the whole crew to use and they take turns playing various games. Unfortunately, a technical glitch in the system caused by a short circuit means that they have to replace the board. Reading through the manual, they find this weird warning. Help them decipher it so they don’t break it.



Steganography

Asteroid Attack (200)

Along the way, the Avengers run into an asteroid belt and Rocket tries his best to avoid damaging the aircraft. Sadly, a stray asteroid slams into the fuel cell compartment of the ship and the entire ship loses power save for the command module. Captain America pings the logs of the ship to find Tony Stark’s manual on how to repair fuel cells but in the damage, the message seems to have been lost. Help recover them: https://drive.google.com/file/d/1aCLa2jDKomUNiUx6bfE_YbHcAGzLsIYX/view?usp=sharing

Location: South of Boot-2 (500)

After recovering, Captain America turns on the navigation guide and finds the location of the asteroid belt within a system called Boot-2. To safely navigate to the location of Thanos, they must first travel to the south of the system in order to reach the space station that sells new fuel cells and items that they could use to repair the ship. Unfortunately, the image on the navigation guide has become corrupted. Help decipher it: https://drive.google.com/file/d/1Z4m_xqzesfa317ecv0-Zeh4TpraRm1q5/view?usp=sharing

Miscellaneous

Thanos (200)

The Avengers have entered orbit around Thanos' planet but before they can enter, Captain Marvel has volunteered to investigate the planet's defenses in the event Thanos has set a trap. As she lands, she detects a strange energy signature originating from beneath her. She realizes Thanos has protected his home with an impenetrable energy shield and to disable it she must hack into the main computer. Knowing that Thanos has a massive ego, he could only use some combination of lowercase letters in his name as the password. But she finds an exposed md5 hash in the database: 58ccec0cbea909fe7901ecc51d30e0c7.

Searching for the Alpha, Bravo and Charlie (300):

Thor decides to devote his time on the ship brooding over his failure to kill Thanos and prevent the death of half of the entire universe. Unwilling to talk to anyone on the ship after taking a vow of silence during the trip, he proceeds to open a book and read but finds a stark clue that could help the Avengers. But there seems to be something within the book itself: https://drive.google.com/file/d/1vOHjGGO1lHBH2LpdmCx8uZwuDGGWdKzc/view?usp=sharing

Piecing the Titles Together (400):

Black Widow creates a list of all of the possible weapons that they could use against Thanos in the event he tries to bring another fight at the Garden. She tallies up the entirety of the weapon stock but discovers that her list has been jumbled. Help take a look at this for her. She might have added something at the beginning of each word.
https://drive.google.com/file/d/1-g_yEmW0PcvvCHTCZtoq9y7Vh_CL42bF/view?usp=sharing

Optimization

Love (100)

As the Avengers approach Thanos, they realize that Dr. Strange has secretly placed several files in their ship as clues. They open the first file and discover a peculiar function along with a list of numbers. However, the list seems incomplete. Help them complete the list:

```
1: 840
2: 1080
3: 1320
...
17: 2808
18: 2856
19: 2970
20: ???
```

C++:
https://drive.google.com/open?id=1QgGMF9JBvjFqr7UYFEZZxu2T9GFvejyP

Java:
https://drive.google.com/file/d/1I-a4u612uwYc1GvAht4J2-aIdmx05AOR/view?usp=sharing

Python:
https://drive.google.com/file/d/1ZX5syDJU_ujt86gh9ZzLFnxRU8gxLjn1/view?usp=sharing

Time (400)

The second file is a program with four functions and this time requires an input. The Avengers discover that multiple positive integers can cause the program to output “We’re in the endgame now”, but they might not have enough time to find all of them! Compute the xor of all possible answers.

C++:
https://drive.google.com/file/d/1gpfWvpHPYIcL_xVmUjGWB0pWQa2NcfV-/view?usp=sharing

Java:
https://drive.google.com/file/d/1z8PbfjbsyltB9GqVZeFxrrK8PpXYl9Mj/view?usp=sharing

Python:
https://drive.google.com/file/d/1rz-Kg8-fz_DgncmEhQN1Is7QwTe2XcLZ/view?usp=sharing

Loop (200)

The third file contains two strings: "theavengerstheavengers...theavengers" (where the string "theavengers" is repeated *n* times), and "thanosthanos...thanos" (where the word "thanos" is repeated *n* times). The file also comes with a hash function, but the map *H* that takes a letter to a number is missing. Dr. Strange wants the Avengers to collide with Thanos, so ideally the two strings should always hash to the same value, regardless of what *H* is. What is the smallest positive integer *n* for which the two strings hash to the same value?

C++:
https://drive.google.com/file/d/1cKfzMGyCcgUdQN-RYONY-6XCk0EEFaK0/view?usp=sharing

Java:
https://drive.google.com/file/d/1VlZik3bD20dHO83eizWrEvuYA7SRM5Pu/view?usp=sharing

Python:
https://drive.google.com/file/d/1B4j1ILnjny8TyaCqu1yS5pSJUde1mUQW/view?usp=sharing

Endgame (200)

The final file is fairly short and requires a single nonnegative integer as input. Of the first 14,000,605 nonnegative integers, only one of them causes the program to output 1. Which is it?

C++:
https://drive.google.com/file/d/1PvDofxXIe22rC_DZBsPTgrGCFOppHEO6/view?usp=sharing

Java:
https://drive.google.com/file/d/1QJlIJeQnBYL_CVIkAoaBOq9hDNxmqqvI/view?usp=sharing

Python:
https://drive.google.com/file/d/1uJGrKMp7F3unfDO51a0uKwzys3cUWxGa/view?usp=sharing

Assembly Language

The Stack Laptop Part A (200)

In the meantime, Thanos is enjoying his free time by playing with his new laptop. This laptop has one stack with unlimited capacity. All elements in the stack can be arbitrarily large signed integers. It does not have any programmer-side registers.
The assembly code of this laptop's processor consists of the following instructions:

- PUSH a: Adds integer a to the top of the stack
- POP: Removes a value from the top of the stack
- DUPLICATE: Makes a copy of the value on top of the stack and push it
- ADD: POPs twice, then PUSH the sum of two popped values
- SUBTRACT: Same as ADD, but with subtract (the value on top will be stack[top-1]-stack[top])
- INCREMENT: POP, add one to the value then PUSH
- DECREMENT: Same as INCREMENT, but subtract one
- LABEL label: Defines a new label
- JUMP label: Jumps to the label
- JUMP_IF_ZERO label: Checks the value at the top of the stack, if it is zero, jumps to the label.
- PUSH_L: Pushes the current length of the stack on top

After the program ends, the value at the top of the stack is the output. If the stack is empty, the output is considered to be null.

Consider the following assembly code. What integer should we push in the first command to make the program output 8,590,000,128?

```
PUSH *;
LABEL START_LOOP_1;
DUPLICATE;
PUSH 1;
SUBTRACT;
JUMP_IF_ZERO END_LOOP_1;
JUMP START_LOOP_1;
LABEL END_LOOP_1;

LABEL START_LOOP_2;
PUSH_L;
PUSH 1;
SUBTRACT;
JUMP_IF_ZERO END_LOOP_2;
POP;
ADD;
JUMP START_LOOP_2;
LABEL END_LOOP_2;
POP;
```

The Stack Laptop Part B (300)

What integers should we push in the instructions marked with * to make the program output 8,590,000,128? Report your answers as a string xy where x is the first number and y is the second number. For example if you think the answer is 123 and 400, report 123400.

```
PUSH 1;
LABEL START_LOOP_1;
DUPLICATE;
DUPLICATE;
ADD;
DUPLICATE;
PUSH *;
DUPLICATE;
ADD;
PUSH 2;
SUBTRACT;
SUBTRACT;
JUMP_IF_ZERO FUNCT;
POP;
```

```
LABEL RETURN;
PUSH_L;
PUSH *;
SUBTRACT;
JUMP_IF_ZERO END_LOOP_1;
POP;
JUMP START_LOOP_1;
JUMP END_LOOP_1;
LABEL FUNCT;
POP;
PUSH 1;
ADD;
DUPLICATE;
PUSH 1;
SUBTRACT;
JUMP RETURN;
LABEL END_LOOP_1;
POP;

LABEL START_LOOP_2;
PUSH_L;
PUSH 1;
SUBTRACT;
JUMP_IF_ZERO END_LOOP_2;
POP;
ADD;
JUMP START_LOOP_2;
LABEL END_LOOP_2;
POP;
```

The Infinity Laptop Part A (200)

After a while, Thanos decides to use his infinity stones to increase the power of his laptop. The laptop now supports two new instructions:

- TIME: Puts back the last value a POP instruction has removed from the stack on top of the stack. Note that it only works for values that were popped with a POP instruction. This instruction requires using the time stone and turning back time.
- SWAP: Swaps the top of the stack with the bottom. This instruction requires using the reality stone.

However, using stones has caused some problems; namely, the PUSH instruction does not work anymore.

Assume n=10 distinct integers were already pushed into the stack. There are many programs that Thanos can write using this assembly language with the two new instructions and without PUSH. However, the values in the stack cannot be whatever he wants. To show this, count the number of all possible states that the stack can have after executing a program. Programs can use all instructions defined in this and the previous questions except PUSH.
Two states of the stack are considered to be different if their values or the position of their values are different. For example, [1, 2, 3, 4], [2, 1, 3, 4], [1, 2, 3], [3, 2, 1] and [] are five different states.

The Infinity Laptop Part B (300)

Worried about his laptop, Thanos hires an electrical engineer to fix it. The engineer is not able to completely fix the PUSH instruction, but adds a register to the laptop to calm Thanos down.
The machine now has one register in addition to its stack. This register can hold one integer value, and values can be moved from/to top of the stack to/from this register using the following commands:

- PUSH_REG: pushes the value of the register to the stack
- POP_TO_REG: pops a value from top of the stack and saves it in the register.

Assume n=10 distinct integers were already pushed into the stack. Count the number of all possible states that the stack can have after executing a program. Programs can use all instructions defined in this and the previous questions except PUSH.