Smyth County Public Schools
2017 Computer Science Competition Coding Problems

The Rules

- There are ten problems with point values ranging from 10 to 35 points. There are 200 total points.
- You can earn partial credit for problems.
- You may use Python (codeskulptor.org or IDLE), Scratch, JavaScript, or Google Sheets to solve the problems.
- When you finish a problem, please submit it to coding@scsb.org, using the team email address supplied to you.
- If you use Scratch to solve the problem, submit your solution by sharing the Scratch code and sending the URL of the project page to coding@scsb.org.
- If you use JavaScript or Python to solve the problem, send your program code to coding@scsb.org as an email attachment.
- If you use Google Sheets to solve a problem, share your Google Sheets document with coding@scsb.org.
- Test your code before submitting it. Once you submit a problem solution to coding@scsb.org, you may not resubmit a solution for that particular problem.
- You may use Google or another search engine to look up formulas, such as the formula for converting between Fahrenheit and Celsius. You may also use Google or another search engine to look up the syntax for specific commands; for example, if you can't remember the syntax to create a custom function in Python, you may look that up.
- You may not use Google or other search engines to look up algorithms or code to solve the problems.
- You may not use code stored on your computer, on Google Drive or any other online site, or on any removable media to solve any of the problems. In other words, you have to solve these problems by starting from scratch.

Tips

- You may assign the problems to different team members. This is a recommended strategy for maximizing your coding points.
- If you have questions about any problem, please send a team member to Terry Hawthorne or John King. They will provide the answer to your question to both teams at the same time.
- We don't expect your to solve all ten of the problems.
- Do your best and have fun!

Problem 1: Collatz Conjecture

The Collatz conjecture is a conjecture in mathematics named after Lothar Collatz. It concerns a sequence defined as follows: start with any positive integer n. Then each term is obtained from the previous term as follows: if the previous term is even, the next term is one half the previous term. Otherwise, the next term is 3 times the previous term plus 1. The conjecture is that no matter what value of n, the sequence will always reach 1.

--Wikipedia, Collatz Conjecture

Input

Any positive integer n: 1, 2, 3,...

Output

Your program should output the result of each computation with a counter, and conclude with the statement: "It takes x iterations for n to resolve to 1."
x is the number of calculations (3n + 1, or n/2) required to resolve n to 1. For example, if n = 5, your program should output something like the following:

```
Testing the Collatz Conjecture with a seed value of 5…
(1,16)
(2, 8)
(3, 4)
(4, 2)
(5, 1)
It took 5 iterations to resolve 5 to one.
```

Scoring

Program Element	Point Value
Program accepts any positive integer >= 1	2
Program runs correctly for any seed value that is a positive integer >= 1.	5
Number of iterations is calculated correctly.	5
Each line of output consists of both the counter and the result of each iteration.	3
Output includes a beginning statement that includes the seed value.	2
Output includes an ending statement that includes the seed value and the total number of iterations required to resolve to one.	3
Total Points:	20

Problem 2: Skip Counting

0, 5, 10, 15, 20, 25…
You learned how to skip count in elementary school. Can you write a program to skip count?

Input

- A start value that can be any integer, positive or negative, or zero
- An end value that can be any integer, positive or negative, or zero, but that is not the same as the start value
- A skip count value that can be any positive integer

Output

startValue, …, <= endValue, where … represents the skip counts between the start value and end value.

Examples

Input	**Output**
startValue = 10; endValue = 30; skipCount = 10	10, 20, 30
startValue = 0; endValue = 55; skipCount = 10	0, 10, 20, 30, 40, 50
startValue = 20; endValue = 5; skipCount = 3	20, 17, 14, 11, 8, 5

Scoring

Program Element	**Point Value**
Program prompts for and accepts startValue, endValue, and skipCount values.	5
Output begins with startValue and ends with the endValue, unless the endValue would be skipped over, as in the second example above. In that case, the last number output should be less than the endValue.	5
When startValue is > endValue, program counts backwards from the startValue to the endValue.	5
Total Points:	15

Problem 3: Drawing Regular Polygons

A regular polygon is a polygon in which each side is the same length and all the internal angles are equal. A square is a regular polygon with four sides and four internal 90 degree angles. An octagon is a regular polygon with eight sides and eight internal 135 degree angles. The formula for determining the size of each angle is: ((n - 2) x 180) / n, where n = number of sides. Your assignment is to write a program that can draw regular polygons with up to 10 sides. After drawing the polygon, the program must display the name of the polygon. For example, if the number of sides = 5, your program must display "pentagon" after completing the drawing.

Input

Positive integer between 3 and 10

Output

A drawing of a regular polygon with the correct number of sides, followed by a display of the name of the polygon. All sides of the polygon must be visible in your display window.

Scoring

Program Element	**Point Value**
Program accepts user input for the number of sides, ranging from 3 to 10.	2
Program will not accept input less than 3 or greater than 10.	3
All polygons are drawn correctly. (1 point deduction for any polygon whose sides are not all visible in the drawing window.)	8
Program outputs name of polygon after drawing it. (Names: triangle, square, pentagon, hexagon, septagon, octagon, nonagon, decagon)	8
Program is able to run repeatedly without having to restart it; that is, user should be able to draw one polygon, then be prompted to draw another without a restart.	4
Total Points:	25

Problem 4: Fibonacci Sequence

The Fibonacci sequence is 0, 1, 1, 2, 3, 5, 8, 13, 21, etc. Each new number in the sequence is the sum of the previous two numbers. Write a program that outputs the first 30 Fibonacci numbers, starting at 0. Your program should compute the sequence programmatically; that is, you will not earn any credit for manually computing the first 30 numbers, then printing them.

Scoring

Program Element	**Point Value**
Program correctly outputs the first 30 numbers in a Fibonacci sequence starting with 0.	20
Total Points:	20

Problem 5: Odd or Even

Write a program that accepts any positive integer and outputs the following:

[input number] is [odd / even].

Examples

Input	Output
7	7 is odd.
2066	2066 is even.

Scoring

Program Element	Point Value
Program correctly identifies odd and even numbers.	5
Output matches the specification given above; for example, 48 is even.	5
Total Points:	10

Problem 6: Capture the Flag

Using a web browser, navigate to www.scsb.org/cs/secret.txt. You will find a text file consisting of a series of binary numbers. Those numbers provide a map to a secret file. To decode the clue, you must convert the binary numbers to decimal. The decimal values represent the ASCII code. Decoding the ASCII will give you the clue you need to find the secret file. After you find the secret file, complete this problem by identifying the person or object pictured in the secret file. Submit both the location of the secret file and the name of the object pictured in the file to earn full points.

Scoring

Program Element	**Point Value**
Your are able to decode the clue.	10
You are able to identify the person or object pictured in the secret file.	5
Total Points:	15

Problem 7: Color Switcher

Write a problem that draws a closed geometric shape of your own choosing on screen. The shape can be a circle, rectangle, or anything of your choosing, as long as it is a closed shape. The shape should be filled with the color white. Prompt the user to enter a color from the following list: red, green, blue, black, gray, yellow, orange, or purple. Change the fill color of your shape to match the color input by the user.

Scoring

Program Element	**Point Value**
Program draws a closed geometric shape filled with white.	5
Program prompts user to input a color the following list of colors: red, green, blue, black, gray, yellow, orange, or purple.	5
Program changes the fill color of the closed geometric shape to match the color input by the user.	10
Total Points:	20

Problem 8: Temperature Convertor

Write a program that can convert between Fahrenheit and Celsius temperatures. You may use Google to look up the conversion formulas.

Input

A floating point number that represents a temperature and a string that represents the temperature scale, either Fahrenheit or Celsius. You may use F for Fahrenheit and C for Celsius.

Output

The equivalent temperature on the other temperature scale.

Examples

Input	**Output**
68 F	20 C
100 C	212 F

Scoring

Program Element	**Point Value**
Program accepts temperature to be converted as a floating point value and temperature scale as C or F.	5
Program correctly converts Fahrenheit to Celsius	5
Program correctly converts Celsius to Fahrenheit	5
Total Points:	15

Problem 9: Parsing the Time

Computer operating systems can manage date and time calculations by representing the particular date/time as the number of seconds, or milliseconds, that have elapsed since a starting date/time, such as midnight on Jan. 1, 1970. To put dates and times into human-friendly format, coders have to learn how to parse a large number of seconds into years, days, hours, minutes, and seconds. Demonstrate your ability to do this by writing a program that can accept a number of seconds ranging from 0 and 9,999,999 and display that number as days, hours, minutes, and seconds.

Examples

Seconds	**Output**
3600	0 day(s), 1 hour(s), 0 minute(s), 0 second(s)
65	0 day(s), 0 hour(s), 1 minute(s), 5 second(s)
9,999,999	115 day(s), 17 hour(s), 46 minute(s), 39 second(s)

Scoring

Program Element	**Point Value**
Program accepts input from 0 to 9,999,999	5
Program demonstrates use of floor division and modulus operations.	10
Program correctly parses a set of test values.	10
Total Points:	25

Problem 10

Code a game that implements this dice game.

- The name of the game is Random Rocks. The program should announce this fact.
- There are two players. The game must prompt the players to enter their names.
- The first player to win three rounds is the winner.
- A round consists of each player rolling a single, six-sided die five times in a row. The score for the round is the sum of the five rolls. For example, if player one rolls 1, 2, 4, 3, and 6, his score for the round is 16. The player with the highest score wins the round. If there is a tie, the round doesn't count.
- Important Note: Should a player roll the same value on all five rolls, he earns a 20-point bonus for that round.
- At the end of each round, the game should display the total roll for each player in that round, the name of the winning player, and the overall score. For example, if player one's total is 17 and player two's total is 19, the program should display the following after the first round:

```
Round 1 Results
Player 1: 17; Player 2: 19
Player 2 wins the round.
Player 2 leads 1-0.
```

If there is an overall tie at the end of a round, the program should display:

```
The game is tied at 1-1 or 2-2.
```

When one player wins three rounds, the program should display, in addition to the round summary, the following statement:

```
Player [1 / 2] is the winner!
```

Scoring

Program Element	Point Value
Program announces the name of the game.	1
Program explains the rules of the game accurately.	2
Program prompts for the name of each player and stores that name in a variable for later use.	4
Program uses random number generator to simulate the rolling of the die.	1
Program accurately sums each player's round total.	2
Program accurately determines the winner of each round.	5
Program summarizes the results of each round as shown in the game specification.	10
Program determines the winner and displays the winner as shown in the game specification.	10
Total Points:	35