Name:
Date:

AP Computer Science A
Unit 6 Array Programs

1. Write a method that returns an array filled with the first n Fibonacci numbers. The first element should be $F_0 = 0$, the second element should be $F_1 = 1$; each subsequent element should be equal to the sum of the two previous ones. For example, `fibonacci (6)` should return an array with seven elements: 0, 1, 1, 2, 3, 5, 8.

2. If you take any two positive integers m and n $(m > n)$, then the numbers a, b, and c where

   $a = m^2 - n^2$ $\quad b = 2mn$ $\quad c = m^2 + n^2$

   form a Pythagorean triple:

   $$a^2 + b^2 = c^2$$

   You can use algebra to prove that this is always true.

   Write a method `makePythagoreanTriple` that takes two integer parameters, m and n, swaps them if necessary to make $m > n$, calculates the Pythagorean triple using the above expressions, swaps a and b, if necessary, to make $a < b$, places the resulting values a, b, and c into a new array of three elemetns and returns that array. Test your method in a simple program.

3. In SCRABBLE, different letters are assigned different numbers of points:

A – 1	E – 1	I – 1	M – 3	Q – 10	U – 1	X – 8
B – 3	F – 4	J – 8	N – 1	R – 1	V – 4	Y – 4
C – 3	G – 2	K – 5	O – 1	S – 1	W – 4	Z – 10
D – 2	H – 4	L – 1	P – 3	T – 1		

   Write a method `computeScore (String word)` that returns the score for a word without using any `if` or `switch` statements. (Hint: find the position of a given letter in the alphabet string by calling `indexOf`; get the score for that letter from the array of point values, and add to the total.

Name _____
Date _____

AP Computer Science A
Unit 6 For Each Loop Programs

1. Declare a String array. Use an initializer list to specify 5 values.

2. Print the array using a for loop.

3. Print the array using a for each loop.

4. Print the contents of the array in reverse using a for loop. [Don't move them – that's in the next exercise.]

5. Reverse the contents of the String array. Do this without declaring a new array. Swap the first and last element, etc. Use a for loop that can handle any size array.

6. Print the modified array using a for-each loop.

7. Declare an int array to hold 10 values. Use a loop to initialize each element to a random integer in the range [1, 50].

8. Print the int array using a for-each loop.

9. Using for each loops, find the following for your int array. Print each value after the loop(s) execute(s).
    1. The sum and average of the elements
    2. The minimum value
    3. The number of elements with a value greater than 25

10. Declare a boolean array to hold 10 values. Use a loop to initialize the elements to alternating values: false, true, false, true...

11. Print the boolean array using a for-each loop.

Name:
Date:

APCS Unit 6 Array Programs B

More Array Practice Programs

1. Create an array that can hold ten integers, and fill each slot with a different random value from 1-50.
   Display those values on the screen, and then prompt the user for an integer. Search through the array, and count the number of times the item is found.

2. **Hangman Program**
   You are to write a program that will:
   - Read in a word from the user (while the opposing player turns head). Store this word in a String called word.
   - Create a boolean array of size word.length() – You will use the boolean array to keep track of which letters should be displayed. Initially, this array should be initialized as false (this will happen by default).
   - Create a constant (final) variable that corresponds to the total number of misses a user can have before he is hung.
   - Create a variable to store the number of times a user has guessed incorrectly.
   - Create a loop that will:
     1. Display the puzzle. For example: if the word is "Hello", we could display the puzzle to the user as _ _ _ _ _. As letters are guessed correctly, it will begin looking like _ e _ _ o. You will do this by cycling through the boolean array. If the value at a given index is false, you will display a "_" to the screen, if that value is true, you will instead display the corresponding letter in the word (whatever is at the same index).
     2. Prompt the user to enter a letter.
     3. Check to see if this letter is in the word.
        a. If it is, flip the boolean value in the boolean array that corresponds to the same index as that letter in the word.
        b. If it is not, increment the number of times the user has guessed wrong.
     4. Check to see if the user is 'hung' – if number of incorrect guesses is greater than the constant variable that corresponds to the total number of misses a user can have before being hung.
     5. Check to see if the user has solved the puzzle -- if so, end the game and tell the user he won!
   - If the user becomes hung, inform the user of the solution to the puzzle.
   - For added fun:

- Make sure that a user is not penalized for inputting a capital letter instead of a lowercase letter, or vice versa.
- Keep track of which letters have already been guessed and check to make sure that a user does not ask for the same letter twice.

Something Like War

You are going to create a card game that plays kind of like the game of war.

1. You should begin by creating two arrays of size 26: one array to store player 1's cards and a second to store player 2's cards.

2. Initialize each spot in each array to a random number 2-14, where 2 is a card with the value of 2, 3 a card with the value of 3, … 11 a card with the value of Jack, 12 Queen, 13 King, and 14 Ace.

3. **Self-check:** Print out a list of the cards for each player using a for-loop so that it prints something like:

   Player 1 has [11, 12, 14, 14, 3, 2, 4, 8, 9 10, …]
   Player 2 has [12, 14, 13, 14, 2, 7, 5, 3, 8, 12, …]

   We are printing the cards only so that we can check that the rest of the game plays correctly and that the spots were initialized correctly. At the end of the game, we will remove this, so that the user doesn't know what the card are as the beginning.

4. Cycle through the indices 0 – 25 and compare which player has a higher card value for each "hand" (index). You should keep a tally of how many 'hands' each player wins.

5. For example, if we use the arrays above, in the first 'hand' (at the first index – index 0), player 2 would win the hand because he has a 12 when player 1 has a 11. For the hands shown, player 1 would score 4 hands, player 2 would score 5, and there would be 1 tie.

6. Print out how many wins each player has and how many ties were made.

Additionally:

- Show each hand one by one and include how many points have been earned by each player up to that time.

  For example, for each hand:

```
Player 1          11
Player 2          12
Player 2 wins!!

Player 1: 0
Player 2: 1
Ties: 0
```

- Display J, Q, K, and A instead of 11, 12, 13, 14 respectively when printing out hands and the initial array to the screen.