Anisha Aggarwal

CS161

**Chapter 6 Review Questions (1-6, 8, 16, 19)**

1. A subscript is a(n) an element in the array.

2. Each element in an array must have the same data type as the others.

3. Suppose that you have declared a numeric array named values that has 13 elements. Which of the following must be true? cd

4. The subscripts of an array are always integers.

5. Suppose that you have declared a numeric array named numbers, and two of its elements are numbers[1] and numbers[4]. You know that there are exactly two elements between those two elements.

6. Suppose that you have declared a numeric array named numbers, and two of its elements are numbers[1] and numbers [4]. You know that the array has atleast five elements.

8. A program contains a seven-element array that holds the names of the days of the week. At the start of the program, you display the day names using a subscript named dayNum. You display the same array values again at the end of the program, where you can use dayNum, but can also use another variable as a subscript to the array.

16. Each element in a seven-element array can hold one value(s).

19.In every array, a subscript is out of bounds when it is negative.

**Chapter 6 Programming Exercises**

**2a. Algorithm**

1. Create all the variables necessary
   1. int array[12]; int sum = 0; int average = 0;
2. Make a loop that will:
   1. read in the user’s integers (up to 12 integers)
   2. calculate the new sum each time an integer is added
   3. print out each of the new integers added
3. Once the loop is over, calculate the average by dividing the sum by 12 and print it out

**Pseudo Code**

1. Create an integer array of 12 elements
2. Create a variable that will add up all of the integers the user has entered
3. Create a variable that will hold the average of each entered number
4. Start a for loop that will read in the user’s input for up to 12 elements of the array
   1. As each element is read into, add the input to the sum variable
   2. After each element is read into, print each element
5. Once all of the elements are read into, calculate the average by dividing the sum by 12

**2b. Algorithm**

1. Create all the variables necessary
   1. int array[12]; int sum = 0; int average = 0; int count = 0; char add\_more = ‘N’;
2. Make a loop that will:
   1. read in the user’s integers (up to 12 integers)
   2. calculate the new sum each time an integer is added
   3. add one to the count
   4. print out each of the new integers added
   5. and ask whether the user wants to add more integers or not
3. Once the loop is over, calculate the average by dividing the sum by the count

**Pseudo Code**

1. Create an integer array of 12 elements
2. Create a variable that will add up all of the integers the user has entered
3. Create a variable that will hold the average of each entered number
4. Add an integer variable that will count the number of integers being added that is initialized to 0
5. Create a char variable that will hold whether the user wants to add another integer or not called add\_more
6. Start a do-while loop that will get user’s input for less than 12 elements of the array and will keep reading while add\_more is yes
   1. Ask the user to input an integer that will be added into array[count]
   2. As each element is read into, add the input to the sum variable
   3. Increase the count by 1
   4. Ask the user if they would like to add another integer
   5. After each element is read into, print each element
7. Once either all of the elements are read into and/or add\_more is no, calculate the average by dividing the sum by 12 and print it out

**3a. Algorithm**

1. Create all the variables necessary
   1. int array[12]; int largest\_num = 0; int smallest\_num = 0;
2. Make a loop that will:
   1. read in the user’s integers (up to 12 integers)
   2. check if the number entered is larger than largest\_num – if it is then make that the new largest\_num
   3. check if the number entered is larger than smallest\_num – if it is then make that the new smallest\_num
   4. print out each number entered
3. Once the loop is over, print out the largest*\_*num and smallest\_num

**Pseudo** **Code**

1. Create an integer array of 12 elements
2. Create a variable that will add up all of the integers the user has entered
3. Create a variable that will hold the largest number initialized to 0
4. Create a variable that will hold the smallest number initialized to 0
5. Start a for loop that will read in the user’s input for up to 12 elements of the array
   1. As each element is read into checks to see if that input is greater than largest\_num
      1. if it is, set that number to largest\_num
   2. As each element is read into checks to see if that input is smaller than smallest\_num
      1. if it is, set that number to smallest\_num
   3. After each element is read into, print each element
6. Once all of the elements are read into, print out the smallest\_num and the largest\_num

**Hangman Changes**

If there were multiple turns alternating players, there would have to be a deciding factor on when to stop switching turns. This can be determined with either a while loop that will stop when a player gets a certain number of points or there can be a do-while loop that stops when the user says they don’t want to play anymore. For the first option there would be a function that would create a point system so that when the player guessing guesses correctly, they get points. For the second option, there would be a char variable that will first prompt the user whether they want to play more or not and if they answer is ‘Y’ then they continue, but if it’s ‘N’ they will stop.