1. **Generics:**
2. Which of the following will compile assuming E is unbounded?
3. E temp;
4. E temp = new E();
5. E[] arr = new E[10];
6. E[] arr = (E[]) new Object[10];
7. temp instanceOf T;
8. Only I
9. I and II
10. **I and IV**
11. I, III, and V
12. All of the above
13. True/False: You can pass a primitive type in as a type parameter.
    1. True
    2. **False**
14. **Searching:**
15. Given the following input array, show the elements at each step of binary search for the element. If there is part of the array that is not being considered during a particular step, don’t write anything in those places. Use integer division to decide where to split.

Input:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 12 | 14 | 15 |

**Search for: 9**

Step 1:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **8** | **9** | **12** | **14** | **15** |

Step 2:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **8** | **9** |  |  |  |

Step 3:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | **9** |  |  |  |

1. **Asymptotics:**
2. What is the **best** Big O approximation for binary search?
   1. O(n)
   2. **O(log n)**
   3. O(n^2)
   4. O(2n)
3. What is the **best** Big O approximation for the following method?

public int sumThree(int[] arr) {

int total = 0;

for (int i = 0; i < 3; i++) {

total += arr[i];

}

return total;

}

1. **O(1)**
2. O(n)
3. O(log n)
4. O(n^2)
5. What is the **best** Big O approximation for the following method?

public boolean containsName(String[][] arr, String name) {

boolean found = false;

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr[i].length; j++) {

if (arr[i][j].equals(name)) {

found = true;

}

}

}

return found;

}

1. O(n)
2. O(2n)
3. O(n log n)
4. O(n^2)
5. What is the **best** Big O approximation for the following method?

public int middleValue(int[] arr) {

int min = arr[0];

for (int i = 0; i < arr.length; i++) {

if (arr[i] < min) {

min = arr[i];

}

}

int max = arr[0];

for (int j = 0; j < arr.length; j++) {

if (arr[j] > max) {

max = arr[j];

}

}

return (min + max) / 2;

}

1. O(n)
2. O(2n)
3. O(1)
4. O(n^2)
5. What is the **best** Big O approximation of the following method?

public int partialSum(int[] arr) {

int total = 0;

for (int i = 0; i < arr.length; i \*= 2) {

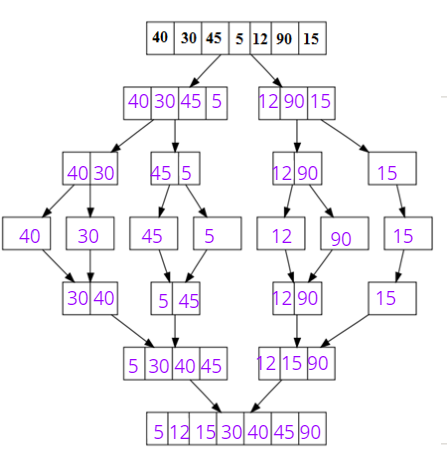
total += arr[i];

}

return total;

}

1. O(n)
2. O(n / 2)
3. O(log n)
4. O(n^2)
5. **Sorting:**
6. Given the following array, show the elements after each step of merge sort:



1. How many passes would it take to sort the following array using the **bubble sort** algorithm?

[1, 2, 6, 3, 10, 14]

1. 1
2. **2**
3. 4
4. 6
5. Which interface must be implemented by a class in order to use .sort() on a List of its type?
   1. Comparator<T>
   2. Sortable<T>
   3. Comparable<T>
   4. Ordered<T>
6. **Recursion**
7. What is the function of the following recursive method?

public String mysteryMethod(String str) {

if (str.length() == 1) {

return str;

}

return mysteryMethod(str.substring(1)) + str.charAt(0);

}

1. It returns the first character of the passed in String
2. It reverses the passed in String
3. It counts the number of characters in the passed in String
4. It returns the passed in String
5. What parts are required in a recursive method?
6. A base case
7. A recursive call
8. Progress toward the base case
9. All of the above
10. **Exceptions**
11. Which of the following method headers is correctly throwing a FileNotFoundException?
12. public FileNotFoundException errorMethod()
13. public void errorMethod() throw FileNotFoundException
14. public void errorMethod() new FileNotFoundException
15. public void errorMethod() throws FileNotFoundException
16. \_\_\_\_\_\_\_\_\_\_ Exceptions must be handled for your code to compile.
    1. Checked
    2. Unchecked
17. True/False: The finally block will always execute, even if the try block throws an Exception.
    1. True
    2. False