EECS1022 Summer 2019 Test 3 Version B

Q1: Examine the fragment shown below in which `m` is an initialized variable of type `int` and `list` refers to an empty list of type `ArrayList<Integer>`. Given that the fragment's output is 13, what is the initial value of the variable `m`?

```
list.add(5); list.add(11);
int p = list.get(1) - list.get(0);
System.out.println(m + p);
```

Write the answer without adding anything to it (such as extra quotes, leading or trailing text, = signs, ...). Write XXX in case of errors.

Q2: Examine the fragment shown below in which `m` is an initialized variable of type `int` and `set` refers to an empty set of type `TreeSet<Integer>`. Given that the fragment's output is 10, what is the initial value of the variable `m`?

```
set.add(45); set.add(24); set.add(53);
boolean b = set.add(24);
if (b)
{
     System.out.println(m - set.size());
}
else
{
     System.out.println(m + set.size());
}
```

Write the answer without adding anything to it (such as extra quotes, leading or trailing text, = signs, ...). Write XXX in case of errors.

Q3: Examine the fragment shown below in which `m` is an initialized variable of type `int` and `map` refers to an empty map of type

`TreeMap<Integer,Integer>`. Given that the fragment's output is 7, what is the initial value of the variable `m`?

```
map.put(3, 2); map.put(3, m); map.put(9, 8);
if (!map.containsKey(6))
{
     System.out.println(map.get(9) - map.get(3));
}
else
{
     System.out.println(map.get(3) + map.get(9));
}
```

Write the answer without adding anything to it (such as extra quotes, leading or trailing text, = signs, ...). Write XXX in case of errors.

Q4: Which of the following statements about a `Map<Integer,String>` is *false*?

- ○ Such a map contains a set of key-value pairs *(i,s)* where *i* is an integer and *s* is a string.
- ○ For every key in the map, there is a unique value that it associates to that key.
- ○ For every value in the map, there is a unique key that it associates to that value.
- ○ The collection of all the keys of the map is a set.
- ○ none of the above

Q5: Which of the following statements is *false*?

- ○ Both `ArrayList<String>` and `LinkedList<String>` implement all the methods in the `List<String>` interface.
- ○ `ArrayList<String>` and `LinkedList<String>` use different representations of lists of strings.
- ○ If we create an empty `ArrayList<String>` and an empty `LinkedList<String>` and add the same 3 strings to them, `equals` will return true when called to compare the resulting lists.
- ○ If we create an empty `ArrayList<String>` and an empty `LinkedList<String>` and add the same 3 strings to them, `toString()` will return the same result for both lists.
- ○ none of the above

Q6: Given two non-null, non-empty sets `a` and `b`, the method below returns the set of:

```
public static Set<String> work(Set<String> a, Set<String> b)
{
     Set<String> s = new HashSet<String>();
     for (String e : b)
     {
          if (!a.contains(e))
          {
               s.add(e);
          }
```

```
        }
        return s;
}
```

- ○ elements in a but not in b
- ○ elements in b but not in a
- ○ elements in the intersection of a and b
- ○ elements in the union of a and b
- ○ none of the above

Q7: The method below returns:

```
public static int m(Map<Integer,Integer> map)
{
        int edge = Integer.MIN_VALUE;
        for (int key : map.keySet())
        {
                if (map.get(key) > edge)
                {
                        edge = map.get(key);
                }
        }
        return edge;
}
```

- ○ the largest value in the map
- ○ the largest key in the map
- ○ the smallest value in the map
- ○ the smallest key in the map
- ○ none of the above

Q8: The worst-case complexity of the method below is (`N` is the size of the collection):

```
public static int meth(HashSet<String> set1, HashSet<String> set2)
{
        int result = 0;
        for (String a : set1)
        {
                if (set2.contains(a))
                {
                        result++;
                }
        }
        return result;
}
```

- ○ O(1)
- ○ O(N)
- ○ O(NlgN)
- ○ O(N^2)
- ○ none of the above

Q9: Which of the following is the set of all strings that match the regular expression `^(((ab)?[cd])|e)f$` ?

- ○ { "acf", "adf", "abcf", "abdf", "ef" }
- ○ { "acef", "adef", "abcef", "abdef" }
- ○ { "ac|ef", "ad|ef", "abc|ef", "abd|ef" }
- ○ { "cf", "df", "abcf", "abdf", "ef" }
- ○ none of the above

Q10: Which of the following strings does *not* contain a match for the regular expression `ab*c$` ?

- ○ "ac"
- ○ "cabc"
- ○ "abbc"
- ○ "abcc"
- ○ none of the above

Q11: Implement the method below which takes a set of positive integers `set`, and returns the product of all the elements in the set.

For example, if `set` has the value `[2, 3, 5]`, then the method returns the value `30`; if the set is empty, the method returns 1.

Note: After you develop and test, copy and paste the indicated method only, starting with the shown header and ending with the closing brace. Your code must compile when placed in a class that has only two imports at its top:

`import java.util.regex.*;`
`import java.util.*;`

```
public static int prod(Set<Integer> set)
{

}
```

Q12: Implement the method below which takes a non-negative integer `k` and a list of strings `list`, and returns the list of all the elements in `list` whose length is equal to `k` (if there are no elements in `list` that are of length `k`, it returns the empty list). The argument `list` should not be changed by the method.

For example, if `k` has the value `3` and `list` has the value `[john, ann, paul, ted, helen]`, then the method returns the value `[ann, ted]`.

Note: After you develop and test, copy and paste the indicated method only, starting with the shown header and ending with the closing brace. Your code must compile when placed in a class that has only two imports at its top:
`import java.util.regex.*;`
`import java.util.*;`

```
public static List<String> select(int k, List<String> list)
{

}
```

Q13: Implement the method below which takes a string `s`, and returns a list of all the flight number/ID substrings in `s`; if there is no flight number in `s`, it returns an empty list.

A flight number/ID is a string of 2 upper case letters, followed by an optional space, followed by 3 or 4 digits.

For example, when `s` has the value `"Flights AC 890 and WS2504 leave at 9:30. AC620, D7055, and UA 8454 arrive at 10:50. TS78 is delayed."`, the returned value should be `[AC 890, WS2504, AC620, UA 8454]`.

Note: After you develop and test, copy and paste the indicated method only, starting with the shown header and ending with the closing brace. Your code must compile when placed in a class that has only two imports at its top:
`import java.util.regex.*;`
`import java.util.*;`

```
public static List<String>   findAllFlightIDs(String s)
{

}
```

Q14: Implement the method below which takes a string `s` and a map from integers to strings `map`, and returns a set containing all the values in `map` that come after `s` in the lexicographic order.

For example, if `s` has the value `"john"` and `map` has the value `{12=john,15=ann, 17=paul, 8=ted, 21=helen}`, then the method returns the value `[paul, ted]`.

Note: After you develop and test, copy and paste the indicated method only, starting with the shown header and ending with the closing brace. Your code must compile when placed in a class that has only two imports at its top:
`import java.util.regex.*;`
`import java.util.*;`

```
public static Set<String>  filter(String s, Map<Integer,String> map)
{

}
```

Q15: Implement the method below which takes a map from strings to sets of strings `m1` and a map from strings to sets of strings `m2`, and returns a new map that is the union of `m1` and `m2`, that is, whose keys are those that are either keys of `m1` or keys of `m2`, and such that the value of a key is the union of its value under `m1` and its value under `m2`.

For example, if `m1` is the map `{AA=[aa, bb, cc], BB=[ee, ff]}` and `m2` is the map `{AA=[bb, dd], CC=[ee, gg]}`, the returned value should be `{AA=[aa, bb, cc, dd], BB=[ee, ff], CC=[ee, gg]}`.

The method should not make any modification to its arguments `m1` and `m2`.

Note: After you develop and test, copy and paste the indicated method only, starting with the shown header and ending with the closing brace. Your code must compile when placed in a class that has only two imports at its top:
`import java.util.regex.*;`
`import java.util.*;`

```
public static Map<String, Set<String>> mapUnion(Map<String, Set<String>>m1, Map<String,Set<String>> m2)
{

}
```

Logout
