1 Quickies

Please explain what the output for each code segment will be (Compile Time Error, Run Time Error, Output) and *why*. Assume that each code segment is in a main function and the necessary library files are included if not otherwise noted.

1.1

```
String s1 = "Hello World";
String s2 = "Hello ";
s2 = s2 + "World";

Point a = new Point(3, 4);
Point b = new Point(3, 4);

if (s1 == s2)
  System.out.println("same strings");
if (a == b)
  System.out.println("same points");
if (a.equals(b))
  System.out.println("equal points");
```

Output: Compile Time Error, Run Time Error, or Output (please specify)

Ans:
equal points

1.2

```
public class Test {

  int val;
  String name;

  public Test () {
    val = 10;
    name = "61b student";
  }

  public void report () {
    System.out.printf("Hi, my name is %s, I got a %h on the test", name, val);
  }

  public static void main (String[] args) {
    Test aa = new Test();

    report();
  }
}
```

Output: Compile Time Error, Run Time Error, or Output (please specify)

Ans:
Compile Time Error, report is not static

1.3

```
class NamedObj {
  String name;

  public NamedObj (String name) {
    this.name = name;
  }

  public static void main (String[] args) {
    Object p1 = new NamedObj("p1");
    Object p2 = p1;

    if (p1 == p2)
      System.out.println("Same Object");
  }
}
```

Output: Compile Time Error, Run Time Error, or Output (please specify)

Ans:
Same Object

1.4

```
class NamedObj {
  String name;

  public NamedObj (String name) {
    this.name = name;
  }

  public static void main (String[] args) {
    Object p1 = new NamedObj("p1");
    Object p2 = new NamedObj("p1");

    if (p1 == p2)
      System.out.println("Same Object");
  }
}
```

Output: Compile Time Error, Run Time Error, or Output (please specify)
Ans:
<bnothing>

1.5

```
int x1, x2;
x1 = 5;
x2 = 7;

x1 = x1 ^ x2;
x2 = x1 ^ x2;
```

```
x1 = x1 ^ x2;

System.out.println(x1 + x2);
System.out.println(x1 + " " + x2);
```

Output: Compile Time Error, Run Time Error, or Output (please specify)

Ans:

12
7 5

2 Getting to the Finish

We provide you with the following declarations and methods that do *exactly* what the comments describe.

```
final static int UP = 0, DOWN = 1, LEFT = 2, RIGHT = 3;

public class MazeNode {
  //private fields
  <omitted>

  //methods

  public String getName() {
  //returns the name of this MazeNode
  }

  public void markNode () {
  // marks this node, subsequent calls to haveVisit() will always return true
  }

  public boolean haveVisit () {
  // returns true iff markNode() has ever been called on this node
  // false otherwise
  }

  public boolean canMove (int dir) {
  // returns true iff if there is a reachable neighbor in direction dir
  // dir must be UP, DOWN, LEFT, or RIGHT, else throws NoSuchDirectionException
  }

  public MazeNode move (int dir) {
  // returns the neighbor of this node in direction dir if such a reachable neighbor exists
  // else throws DeadEndException
  // dir must be UP, DOWN, LEFT, RIGHT, else throws NoSuchDirectionException
  }
}
```

2.1

You are tasked to write a function, that given a MazeNode as a starting point, you must find the least amount of steps (succesive *move(int dir)* calls) needed to get to a MazeNode with a name that includes the phrase "finish" (in any combination of upper/lowercase letters) in it.

```
public int findFinish (MazeNode start) {
//this is most likely not enough space to finish the method
//Solution :

  Queue<MazeNode> fringe = new Queue<MazeNode>;
  int depth  = 0;
  int thisgen = 1;
  int nextgen = 0;
  String = ".*finish.*";
  fringe.offer (start);
  start.markNode();
```

```
  //The idea is to do BFS while keeping track of depth, and terminate once we find a solution

  MazeNode current;

  while (fringe.size() != 0) {
    current = fringe.remove();

    if ()

    thisgen--;
    if (thisgen == 0) {
      thisgen = nextgen;
      nextgen = 0;
      depth++;
    }

    for (int i = 0; i < 4; i++) {
      if (current.canMove(i) && !current.move(i).haveVisit()) {
        fringe.add(current.move(i));
        current.move(i).markNode();
      }
    }


  }

}
```

2.2

Please give an asymptotic run time analysis of your method provided that there are N mazeNodes that are in the connected graph extending from the start node.

Ans: It's BFS. The runtime analysis should be done best-case, worst-case.

3 Family Trees

Given this type of tree structure:

```
public class FamilyTree {
  private FamilyTreeNode myRoot;

  <omitted methods and fields>
}

public class FamilyTreeNode {

  public FamilyTreeNode myParent;
  public ArrayList<FamilyTreeNode> myChildren;
  public String myName;
```

```
}
```

We want to implement the following method in FamilyTree class:

```
public string nameOfClosestCommonAncestor(String name1, String name2) {
  FamilyTreeNode f1 = myRoot.search(name1);
  FamilyTreeNode f2 = myRoot.search(name2);

  FamilyTreeNode ans = closestCommonAncestor (f1, f2);

  return ans.myName;
}
```

3.1

Please help implement the search function in FamilyTreeNode class:

```
public FamilyTreeNode search (String name) {
//looks through this node and all its children and descendants
//for a node with myName equal to name
//returns this node if found, null if no such node exists

//Solution
if (myName.equals(name))
  return this;

for (FamilyTreeNode child : myChildren) {
  FamilyTreeNode temp = child.search(name);
  if (temp != null)
    return temp;
}

return null;
}
```

What is the runtime of this function?

Ans:

Theta(number of nodes under this)

3.2

Please help implement the closestCommonAncestor function in FamilyTree class:

```
private static FamilyTreeNode closeCommonAncestor (FamilyTreeNode f1, FamilyTreeNode f2) {
//looks for the closest common ancestor of f1 and f2.
//The closest common ancestor of two nodes f1 and f2 is the closest node  the
//furthest from the root  thats an ancestor of both f1 and f2. An ancestor being
//itself, its parents, grandparents and so on.

//Solution
Hashset<FamilyTreeNode> hash = new Hashset<FamilyTreeNode>();
while (f1 != null) {
  hash.add(f1);
  f1 = f1.parent;
}
while (f2 != null) {
```

```
  if (hash.contains(f2))
    return f2;
  f2 = f2.parent;
}
retrun null;
}
```

What is the runtime of this function?
Ans:
Theta(depth of f1 + depth of f2)