Midterm 1 Study Guide

CSC 210 Fall 2024

In addition to coding questions (see practice problems 1 and 2), there will be questions about concepts in Midterm 1. This guide is to help you prepare for questions about concepts.

Data Structures

1. A game developer wants to create an inventory that shows how many of each item a player has collected in the game. What would be the best data structure to use to accomplish this?
2. A local bakery owner wants to have a system where orders are stored according to the order they are placed (first come, first served). What would be the best data structure to use to accomplish this?
3. A school principal wants to create an attendance for family members of students coming to school events. Only the student name is recorded once, but each student might have multiple family members. What would be the best data structure to use to accomplish this?

Reading Code

1. Code understanding (string manipulation): Given the below code, what is its output?

```
String csv_input = "Jane Doe, 1234567, 3.9";
String[] fields = csv_input.split(",");

System.out.println(fields[0]);
System.out.println(fields[0].indexOf("e"));
System.out.println(fields[2]);
System.out.println(fields[2].indexOf("."));
```

```
System.out.println(fields[1] == " 1234567");
System.out.println(fields[1].equals(" 1234567"));
```

2. Code understanding (array manipulation): Given the below code, what is its output?

```
ArrayList<Integer> nums = new ArrayList<Integer>();
nums.add(99); nums.add(98); nums.add(97);

for (int i : nums) System.out.println("i="+i);

for (int j=0; j<nums.size(); j++) {
  System.out.println("j="+j);
  System.out.println("nums[j] = "+nums.get(j));
}
```

3. Code understanding (recursion): Given the below code, what is its output?

```
import java.util.ArrayList;

public class RecursiveFoo {

  public static void main(String[] args) {
        ArrayList<Integer> result = new ArrayList<Integer>();
        foo(5, result);
        for (int r : result) System.out.println(r);
    }

    public static int foo(int n, ArrayList<Integer> soFar) {
        if (n == 0) {
            soFar.add(1);
            return 1;
        }
        else {
            int result = n * foo(n-1, soFar);
            soFar.add(result);
            return result;
        }

    }

}
```

4. Code understanding (backtracking): Given the below code, what is its output?

```
import java.util.ArrayList;

public class DiceSum {

    public static void main(String[] args) {
        ArrayList<ArrayList<Integer>> result = new ArrayList<ArrayList<Integer>>();
        ArrayList<Integer> values = new ArrayList<Integer>();
        int targetSum = 5;
        getValidCombinations(2, 0, result, values, targetSum);
        for (ArrayList<Integer> v : result) System.out.println(v);
    }

    public static void getValidCombinations(int count, int currentSum,
                                            ArrayList<ArrayList<Integer>> result,
                                            ArrayList<Integer> values,
                                            int targetSum) {
        if (count == 0) {
            if (currentSum == targetSum) {
                ArrayList<Integer> values2Add = new ArrayList<Integer>(values);
                result.add(values2Add);
            }
        } else if (currentSum + 1 * count > targetSum ||
                   currentSum + 6 * count < targetSum) {
                      return;
        } else {
            for (int v = 1; v <= 6; v++) {
                values.add(v);
                getValidCombinations(count-1, currentSum+v,
                                     result, values, targetSum);
                values.remove(values.size() - 1);
            }
        }
    }
}
```

ANSWERS

Data Structures

1. HashMap
2. ArrayList (or array)
3. HashSet

Reading Code

1. Here's the answer for what the code ouputs:

```
Jane Doe
3
 3.9
2
false
true
```

2. Here's the answer for what the code ouputs:

```
i=99
i=98
i=97
j=0
nums[j] = 99
j=1
nums[j] = 98
j=2
nums[j] = 97
```

3. Here's the answer for what the code ouputs:

```
1
1
2
6
24
120
```

4. Here's the answer for what the code ouputs:

```
[1, 4]
[2, 3]
[3, 2]
[4, 1]
```