CS 211 Computers and Programming Handout n
Fall 2002

Grading Guide for Prelim I

October 17, 2002

NAME:_____

CU ID:_____

Recitation instructor_____

You have one and a half hours to do this exam.

All programs in this exam must be written in Java.

****

Problem	Score
1	
2	
3	
4	
5	
6	
Total	

1. (Java) (10 points) Write a program that will print every other argument given on the command line. For example, if the program is executed with the following command line:

   ```
   zero one two three a b c d
   ```

   it should print

   ```
   one
   three
   b
   d
   ```

   ```
   class Problem1{
     public static void main(String[] args) {
       for (int i = 1; i < args.length; i = i+2)
         System.out.println(args[i]);
     }
   }
   ```

   ```
   1 point for correct declaration of class

   1 point for correct declaration of main

   2 points for printing odd command-line parameters

   2 points for skipping over even command-line paramters

   2 points for correct termination of loop

   2 points for correct syntax

   -1 point for args.length()

   -1 for i =< args.length
   ```

2. (Inheritance) (5 points)

Which letters will be printed when the following program is run? Explain your answer in three or four sentences.

```
class MyClass {
 public static void main(String args[]) {
   B b = new C();
   A a = b;
   if (a instanceof A) System.out.println("A");
   if (a instanceof B) System.out.println("B");
   if (a instanceof C) System.out.println("C");
   if (a instanceof D) System.out.println("D");
   if (a instanceof Object) System.out.println("O");

 }
}

class A{}
class B extends A {}
class C extends B {}
class D extends C {}
```

Answer: The program will print "A", "B", "C", "O". This is because every instance of C is also an instance of A and B because A and B are superclasses of C. By default, Object is the superclass of all classes.

Correct answer, no explanation: -2

3. (Sub-typing) (25 points) Scarlett O'Java was not beautiful but men seldom realized it when caught by mastery of sub-typing in Java. Scarlett has written a Java program with classes and interfaces as shown in Figure 1. This figure also shows the header for interface C.

   Answer the following questions with respect to this figure.

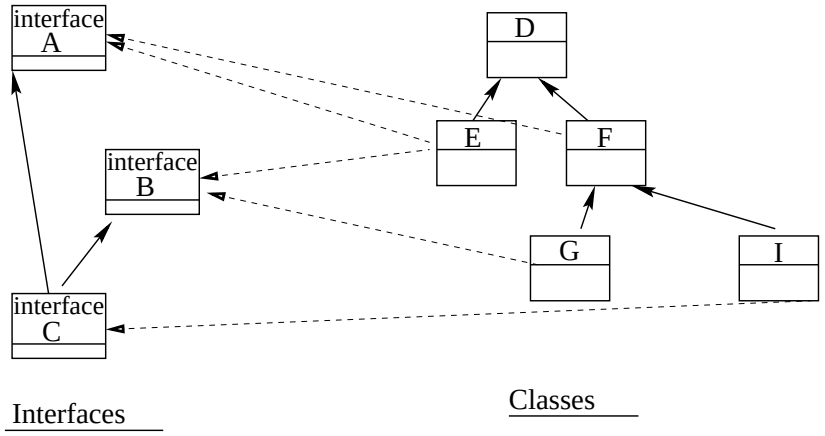
   (a) (4 points) Explain the terms *static method binding* and *dynamic method binding.*

   (b) (2 points) Can the type t_1 of a reference to an object be different from the type t_2 of that object? If so, what relationship exists between t_1 and t_2?

   (c) (1 point) Write the header for class E.

   (d) (2 points) Write two different headers for class G.

   (e) (2 points per statement) Which of the following 8 statements is legal? Explain each answer briefly. No credit will be given unless your explanation makes sense. Each statement should be considered independently of the others.



interface C extends A,B {

...}

Figure 1: Interfaces and Classes for Problem 1

(a) A r = new A();

(b) A r = new F();

(c) A r = (A)(new F());

(d) A r = new G();

(e) F r = new G();

(f) G r = new F();

(g) G r = (G)(new F());

(h) B r = new I();

Answer:

(a) Consider a method invocation "r.m()". In static method binding, the type of reference r determines which method is invoked. Since the type of r is known to the compiler, the method that is invoked can be identified at compile-time. Hence, this is known as static method binding. (Only 2 points for saying "method name resolved at compile-time").

In dynamic method binding, the type of the object referred to by r determines which method is invoked. Since this is known only at runtime, this is known as dynamic method binding. (Only 2 points for saying "method name is bound at runtime").

(b) Yes, t_1 can be different from t_2. However, t_1 is always a supertype of t_2.

(c) `class E extends D implements A,B {..}`

(d) `class G extends F implements B{...}`

`class G extends F implements B,A {...}`

(e) i. A r = new A();
Illegal because A is an interface and cannot be instantiated.

ii. A r = new F();
Legal because class F implements interface A, and upcasting can be implicit in Java.

iii. A r = (A)(new F());
Legal because class F implements interface A.

iv. A r = new G();
Legal because class G extends class F which implements interface A.

v. F r = new G();
Legal because class G is a subclass of class F.

vi. G r = new F();
Illegal because type of reference must always be a supertype of type of object.

vii. G r = (G)(new F());
Illegal because type of reference must always be a supertype of type of object.

viii. B r = new I();
Legal because class I implements interface C which is sub-interface of interface B.

4. (25 points)

   (a) (10 points) Use induction to prove that for all $n > 1$,
   $\frac{1}{n+1} + \frac{1}{n+2} + \ldots + \frac{1}{2n} > \frac{13}{24}$

   (b) (15 points)
   Find all positive integers n for which $2^n > n^2$. Use induction to prove your result.

Answer

(a) (2 points base case, 8 points for inductive case)

Let S(k) = $\frac{1}{k+1} + \frac{1}{k+2} + ... + \frac{1}{2k}$.

Base case: S(2) = 1/3 + 1/4 = 7/12 = 14/24 > 13/24.

Inductive case: assume S(k) > 13/24.

S(k+1) = $\frac{1}{k+2} + \frac{1}{k+3} + ... + \frac{1}{2(k+1)}$

$= S(k) + \frac{1}{2k+1} + \frac{1}{2k+2} - \frac{1}{k+1}$

$= S(k) + \frac{1}{2k+1} - \frac{1}{2k+2}$

$= S(k) + \frac{1}{(2k+1)(2k+2)}$

> 13/24 + some positive quantity

> 13/24

(b)

True for all $n \geq 5$. (3 points)

By substitution, we see that result is true for 1, but not true for n = 2,3,4. (2 points)

Base case (2 points): (n = 5) true because 32 > 25.

Inductive case (8 points): suppose true for k.

Then, $2^{k+1} = 2.2^k > 2.k^2 = k^2 + k^2 > k^2 + 2k + 1 = (k+1)^2$.

For both parts:

```
Base case labeled but wrong: -1
Base case not labeled: -1
```

Inductive step not labeled: -1
Inductive hypothesis wrong: -1
Bad inductive proof : -3

Incorrect algebra : -3

Incomplete proof: -4

For second part:
doesn't say n = 1 works: -1
doesn't show that n = 2,3,4 don't work: -1

5. (Recursion) (20 points) For any positive integer n, let $unos(n)$ be the number of one's in the binary representation of n.

   (a) (3 points) What are the values of $unos(1)$, $unos(2)$, and $unos(3)$?

   (b) (5 points) What is the relationship between $unos(n)$ and $unos(n/2)$ (where / represents integer division)?

   (c) (12 points) Write a public *recursive* class method in Java that takes a positive integer n as a parameter, and returns the value of $unos(n)$.

   Answer:

   (a) (3 points) Values are 1,1,2.

   (b) (1 point) If n is 1, $unos(1) = 1$.
   (2 points) If n is even, $unos(n) = unos(n/2)$.
   (2 points) If n is odd, $unos(n) = unos(n/2) + 1$.

   (c)
```
public static int unos(int n) {

 if (n == 1) return 1;
 if (n/2*2 == n)
    return unos(n/2);
 else return 1 + unos(n/2);

}
```

```
points for public :1
points for static :1
points for return type int :1
points for parameter type :1
points for base case :2
points for odd case :3
points for even case :3

extra base cases: -1
code not recursive: lose shirt
```

6. (Recursion) (15 points) A palindrome is a string that reads the same forwards and backwards. For example, the strings "madam" and "redivider" are palindromes. Write a public recursive class method called isPalindrome that takes a string `s` as a parameter, and returns true if `s` is a palindrome. You may find the following methods in class String useful:

   - s.length(): return the number of characters in string s
   - s.charAt(i): returns the character in position i of string s. In particular, s.charAt(s.length() - 1) returns the last character in the string.
   - s.substring(int start, int end): the substring between positions start and end of string s is returned. For example, if s is the string "Nation", s.substring(3,6) returns the string "ion".

```
public static boolean isPalindrome(String s) {

 if (s.length() < 2) return true;

 if (s.charAt(0) == s.charAt(s.length() - 1))
    return isPalindrome(s.substring(1,s.length() -1));
 else return false;

}
```

```
Base case: 2 points

Inductive case:
check that first and last char's are same: 3 points
recursive call with correct parameters: 10 points


Wrong method signature: -1
No recursion: -10
Bogus recursion: -8
Uses static member to hold data: -5
```

```
Base case fails for s.length() == 0: -1

Inductive case:
charAt() index off by 1: -1
Use equals() with chars: -1
Use [] instead of charAt(): -1
Use == with strings: -1
Wrong arguments to substring(): -3 per mistake
```