

CS162

Computer Science I
Practice Exam 2
Solutions

Fall 2005

1 bonus point on Test 2 to the first person to find each bug in the solutions below and submit the correct answer;

Part 1: Write “if” Statements (20 minutes)

1). Complete the following method that returns `true` if, and only if, the variable `observation` is within `marginOfError` of `target`. For example:

- `isInRange(97, 100, 5)` should return `true`.
- `isInRange(103, 100, 5)` should return `true`.
- `isInRange(100, 100, 5)` should return `true`.
- `isInRange(106, 100, 5)` should return `false`.
- `isInRange(55, 100, 5)` should return `false`.

```
public static boolean isInRange(double observation, double target,
                                double marginOfError)
{

    if (observation <= (target + marginOfError) &&
        observation >= (target - marginOfError))
    {
        return true;
    }
    else
    {
        return false;
    }

} // end isInRange
```

2). Complete the method `getAdmission()` below which determines the admission to a local museum. The prices are as follows:

- Children 3 and under get in free.
- Children 4 through 17 (inclusive) cost \$4.
- College students 18 and under and senior citizens 55 and over cost \$6
- Adults cost \$9.

For example,

- `getAdmission(2, false)` should return `0.0`;
- `getAdmission(2, true)` should return `0.0`;
- `getAdmission(5, false)` should return `4.0`;
- `getAdmission(8, true)` should return `4.0`;
- `getAdmission(22, true)` should return `6.0`;
- `getAdmission(23, false)` should return `9.0`;
- `getAdmission(30, true)` should return `9.0`;
- `getAdmission(45, false)` should return `9.0`;
- `getAdmission(71, true)` should return `6.0`;

```
public static double getAdmission(int age, boolean isStudent)
{
    if (age <= 3) { return 0.0;}

    else if (age >=4 && age <= 17) { return 4.0;}

    else if ((age >= 18 && age <= 27 && isStudent) ||
             age >= 55) { return 6.0;}

    else { return 9.0;}

} // end getAdmission
```

3). In the state of Georgia, you must use a car (or booster) seat until you are either (1) at least 7 years old, or (2) weigh at least 70 pounds. Complete the method `carSeatRequired` below that returns `true` if, and only if, the child described still needs a car or booster seat. For example:

- `carSeatRequired(8, 65)` should return `false`
- `carSeatRequired(8, 75)` should return `false`
- `carSeatRequired(6, 65)` should return `true`
- `carSeatRequired(6, 75)` should return `false`

```
public static boolean carSeatRequired(int age, double weight)
{
    if (age < 7 && weight < 70) { return true;}
    else {return false;}

} // end carSeatRequired
```

Part 2: Write loops

1). Complete the method `multiplesOfX` below that prints all the numbers between `min` and `max` that are multiples of `x`. For example, `multiplesOfX(17, 37, 5)` should print 20 25 30 35

```
public static void multiplesOfX(int min, int max, int x)
{
    int start = min;
    if (start % x > 0)
    {
        start = (min / x + 1) * x;
    }
    for (int loop = start; loop <= max; loop += x)
    {
        System.out.print(loop + " ");
    }

} // end multiples of X
```

2). Complete the method `countHeads` below that flips a coin `numFlips` times, and returns the number of heads. The class `Coin` has only one method: `char flip()`, which randomly returns either 'H' or 'T'. Your task is to count the number of times it returns 'H'.

```
public static int countHeads(Coin c, int numFlips)
{
    int numHeads = 0;
    for (int x = 0; x < numFlips; x++)
    {
        if (c.flip() == 'H')
        {
            numHeads++;
        }
    }
    return numHeads;
}
```

```
} // end countHeads
```

3). Complete the method `vegas` below that simulates playing poker (at a very high level). Count the number of hands played before the player runs out of money. The class `Poker` has only one method: `double changeInMoney()`. This method can return either a positive or negative value representing how much money was won or lost.

```
public static int vegas(double moneyToStart, Poker p)
{
    int handsPlayed = 0;
    double currentMoney = moneyToStart;
    while (currentMoney > 0)
    {
        handsPlayed++;
        currentMoney += p.changeInMoney();
    }
    return handsPlayed;
}
```

```
} // end vegas
```

4). Complete the method `padString` below. This method adds enough spaces to `str` to make it the `desiredLength`. The parameter `endToPad` is either 'R' or 'L', which specifies whether the spaces are to be placed at the beginning or end of `str`. For example, `padString("Dave", 7, 'L')` should return the String " Dave", `padString("John", 7, 'R')` should return the String "John ";

```
public static String padString(String str, int desiredLength,
                               char endToPad)
{
    int numSpaces = desiredLength - str.length();

    String spaces = "";
    for (int x = 0; x < numSpaces; x++)
    {
        spaces += ' ';
    }
    switch (endToPad)
    {
        case 'R': return str + spaces;
        case 'L': return spaces + str;
        default: System.out.println("BIG PROBLEM!!!!"); System.exit(0);
                return "";
    }
}

} // end padString
```

5). (Skip while timing) Complete the method `listPrimes` below, which lists the first `primesDesired` prime numbers. To determine whether a number is prime, use the method `int countFactors(int x)`. Prime numbers have exactly two factors (i.e., an integer `x` is prime if, and only if, `countFactors(x) == 2`). Print only the first `primesDesired` prime numbers. For example, calling `listPrimes(8)` should print `2 3 5 7 11 13 17 19`. (Don't worry; I won't grade the format of your output.)

Special Rule: Because we know that 2 is the only prime number, your method **must test odd integers only** (testing even integers would be a waste of time).

```
public static void listPrimes(int primesDesired)
{
    int primesFound = 1;
    int currentPlace = 3;
    while (primesFound < primesDesired)
    {
        if( countFactors(currentPlace) == 2)
        {
            System.out.print(currentPlace + ' ');
            primesFound++;
        }
        currentPlace+= 2;
    }

} // end listPrimes
```

Part 3: Reading Code

1. Consider the following code:

```
boolean test_boolean(boolean p, boolean q, boolean r)
{
    return !r || p && q;
}
```

Given that `!` has the highest precedence and `||` the lowest, give the return values of

- a. `test_boolean(true, false, true)` **False**
- b. `test_boolean(false, true, false)` **True**

2. (Skip while timing) Given that / has a higher precedence than ==, what is the value of this expression:

$(1 / 4) == 0.25$ **False** $(1 / 4) == 0$ when using integer division

3. (Skip while timing) Consider the following code:

```
int function(int x)
{
    if (x <= 1) { return 1; }
    return x * function(x-1);
}
```

What is the return value of function(5)? **120**

4. Consider the following code:

```
int g(int x)
{
    return x + 2;
}

int h(int x)
{
    return 2 * g(x);
}
```

Compute the return values of:

a. $g(5) = \underline{7}$

c. $g(h(5)) = g(14) = \mathbf{16}$

b. $h(5) = 2 * g(5) = \mathbf{14}$

d. $h(g(5)) = h(7) = 2 * g(7) = \mathbf{18}$

5. Write the output of the following code:

```
int sum = 0;
int count = 0;
while (count < 3)
{
    sum = sum + count*sum;
    count = count + 1;
    System.out.println("Sum is " + sum);
    System.out.println("Count is " + count);
}

while (count < 10)
{
    System.out.println("Now sum is " + sum);
    System.out.println("Now count is " + count);
    sum = sum - count;
    count = count + count;
}
```

```
Sum is 0
Count is 1
Sum is 0
Count is 2
Sum is 0
Count is 3
Now sum is 0
Now count is 3
Now sum is -3
Now count is 6
```

Part 4: Arrays

1). Complete the method `makeCharacterArray` below that takes a `String` as input and returns an array of characters. Each element of the array should contain the corresponding character in the `String`. For example `makeCharacterArray("Summer")` should return a character array in which the first element is 'S', the second element is 'u', the third element is 'm', etc.

```
public static char[] makeCharacterArray(String str)
{
    char[] answer = new char[str.length()];
    for (int x = 0; x < str.length(); x++)
    {
        answer[x] = str.charAt(x);
    }

    return answer;
} // end makeCharacterArray
```

2). Complete the method `getSingleProcessors` below that takes as input an array of `Computer` objects and returns a different array containing only those `Computer` objects that have exactly one CPU. The class `Computer` has only one method `int getNumCPUS()` that returns the number of CPUs. Write your code to handle the case that some of the slots in `allComputers` may be null.

```
public static Computer[] getSingleProcessors(Computer[] allComputers)
{
    Computer[] answer = new Computer[allComputers.length];
    int next = 0;
    for (int x = 0; x < allComputers.length; x++)
    {
        if (allComputers[x] != null &&
            allComputers[x].getNumCPUS() == 1)
        {
            answer[next++] = allComputers[x];
        }
    }
    return answer;
} // end getSingleProcessors
```