**CSCI152 Midterm Exam 2 (Option B).**

**Recursion:**

1. Identify the base case(s) and reduction step of the following recursive function (A, B, and C)…

public static int factorial(int n){

if (n == 0){ **// A**

return 1;

}

else if (n == 1){ **// B**

return 1;

}

else{ **// C**

return n \* factorial(n-1);

}

**A,B are base case(s), C is the reduction step**

***For questions 2-4 classify each implementation of Fibonacci as one of the following…***

1. “Bottom up” dynamic programming solution.
2. “Top down” dynamic programming solution.
3. “Brute Force” recursive solution.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public static long fibi(int n){

if (n == 0) return 0;

if (n == 1) return 1;

return fibi(n-1) + fibi(n-2);

}

**“Brute Force” recursive solution**

public static long fibi(int n){

long[] answers = new long[n+1];

answers[0] = 0;

answers[1] = 1;

for(int i = 2; i <= n; i++){

answers[i] = answers[i - 1] + answers[i - 2];

}

return answers[n];

}

**“Bottom up” dynamic programming solution**

private static int longestFibLong = 92;

private static long[] cache = new long[longestFibLong];

public static long fibi(int n){

if (n == 0) return 0;

if (n == 1) return 1;

if (cache[n] > 0) return cache[n];

cache[n] = fibi(n - 1) + fibi(n - 2);

return cache[n];

}

**“Top down” dynamic programming solution**

1. What are two of the pitfalls of using recursion (that we went over in class)?

**Stack Overflow, Performance Overhead, and Memory Consumption**

**Performance:**

1. **Rank** the following **order of growth functions** from “best” to “worst” in terms of performance (starting at 1)…

|  |  |
| --- | --- |
| **Rank** | **Order of growth function** |
| **4** | n\* log2(n) (linearithmic) |
| **6** | 2n (exponential) |
| **1** | 1 (constant) |
| **3** | n (linear) |
| **5** | n2 (quadratic) |
| **2** | log2(n) (logarithmic) |

1. How many bits are in 1 byte?

**8 bits in 1 byte**

1. Assuming 4 bytes of memory are required to represent a single float value in Java, and an array of any type requires 24 bytes of overhead, how many bytes of memory (total) will be required to store the following array of floats?

float[] floatArray = new float[6];

**4 \* 6 + 24 = 48 bytes**

1. How many bits are in a nibble (this problem is extra credit)?

**4 bits in a nibble**

1. Suppose g(n) = 12n3 + 2n + 10 and g(n) ~ f(n). What is f(n)?

**f(n) = 12n3**

1. Show that the function t = a\*Nb is a line when plotted in log space (hint: take the log of both sides of the equation, and show that the result is the equation of a line)

**t = a\*Nb**

**lg(t) = lg(a\*Nb)**

**lg(t) = lg(a) + lg(Nb)**

**lg(t) = lg(a) + b\*lg(N)**

**Result is the equation of a line with slope = b, and y intercept of lg(a)**

1. Give a linear-time algorithm for reversing a string (i.e., fill in the implementation of the reverse method below)...

public static String reverse(String inputString) {

// Compiling Java code here.

**char[] mysteryString = new char[inputString.length()];**

**int lastIndex = mysteryString.length -1;**

**for (int i = lastIndex; i >= 0; i--){**

**mysteryString[lastIndex - i] = inputString.charAt(i);**

**}**

**return new String(mysteryString);**

}

**Abstract Data Types:**

1. What is the value of c?

String myString = “Hello”;

myString.substring(0,2);

char c = myString.charAt(3);

**c = ‘l’**

1. Which object is “orphaned” after the following code is executed? (Please give your answer as one of “**Object1**” or “**Object2**”).

String myString = new String(“Hello”); **// Object 1**

String myOtherString = new String(“Bye”); **// Object 2**

myOtherString = myString;

**Object 2**

1. One of Java’s most significant features is its ability to automatically manage memory. The idea is to free the programmer from the responsibility of managing memory by keeping track of orphaned objects and returning the memory they use to a pool of free memory. Reclaiming memory in this way is known as?

**Garbage Collection**

1. What does the following code fragment print?

String s = "Hello World";

s = s.toUpperCase();

s.substring(6, 11);

StdOut.println(s);

**“HELLO WORLD”**

1. What is the value of i?

String myString = “how,are,you,doing,today?”;

String[] splitString = myString.split(“,”);

int i = splitString.length;

**Creating Data Types:**

1. True or false, in Java you don’t have to **explicitly** define the default constructor?

**True, in Java the default constructor is available without explicitly defining it**

1. To help enforce encapsulation, what access modifier should be used for instance variables within a class?

**private**

1. By default, every class in Java is a subclass of which superclass (hint: it’s the same word used to describe an instance of a class)?

**Object**

1. For situations where we wish to represent a value from a primitive type as an object, Java supplies built-in reference types known as what?

**Wrapper types**

1. True or false, the ItemPrice interface (below) is a “functional interface”?

public interface ItemPrice{ // What's a functional interface?

public abstract double calculatePrice(double basePrice);

public abstract double someOtherMethod(double basePrice);

}

**False, the ItemPrice interface has more than one abstract method (definition of a functional interface)**

1. Assuming Item is a superclass, what should **KEYWORD** be replaced with in the following class definition for CandyBar?

public class CandyBar **KEYWORD** Item{

private double calories;

public CandyBar(String name, double price, double calories){

super(name, price);

this.calories = calories;

}

}

**Extends**

1. Assuming Animal is an interface, what should **KEYWORD** be replaced with in the following class definition for Cat?

import java.awt.Color;

public class Cat **KEYWORD** Animal{

private String name;

private char sex;

private String dob;

private Color color;

public Cat(String name, char sex, String dob, Color color){

this.name = name;

this.sex = sex;

this.dob = dob;

this.color = color;

}

}

**Implements**

1. Circle the statement that performs “Autoboxing”.

Integer x = 17;  **// A**

int a = x; **// B**

**A**

1. Circle the statement that performs “Unboxing”.

Integer x = 17; **// A**

int a = x; **// B**

**B**