

Computer Science 144  
Allocated Time : 50 minutes

Test 1  
Name: .....

Full marks : 20 (6 bonus points available) Student nr: .....

Important: Please make your answers as detailed as possible by giving a short explanation of your reasoning. For example, simply “yes” or “no” is not acceptable as an answer. The available space at each question gives an indication of the length of the expected answer.

## Question 1 - Strings and chars [9 points]

- a) (1 point) Why is `(s == "")` not a reliable way for checking if `s` is the empty string in Java?

Answer: It compares whether the two strings refer to the same memory location. The two strings can refer to different memory locations which each represent the string "".

- b) (1 point) How can I check whether a string `s` is the empty string in Java?

Answer: Use `(s.equals(""))` or `(s.length() == 0)`.

- c) (1 point) Is there a difference between the empty string and `null` in Java? Explain your answer.

Answer: Yes. The empty string is a string consisting of 0 characters. You can invoke all of the usual string methods, e.g., `length`. On the other hand, `null` is not a string object. You will get a `NullPointerException` if you try to invoke any method on a variable storing `null`.

- d) (1 point) Which one or more of the following code fragments converts all of the strings in the array `a` to upper case?

A)  

```
for (int i = 0; i < a.length; i++) {  
    String s = a[i];  
    s = s.toUpperCase();}
```

B)  

```
for (int i = 0; i < a.length; i++) {  
    a[i].toUpperCase();}
```

C)

```
for (int i = 0; i < a.length; i++) {  
    a[i] = a[i].toUpperCase();}
```

Answer: only C.

- e) (1 point) What does the following recursive function return, given an input string `s`?

```
public static String mystery(String s) {  
    int N = s.length();  
    if (N <= 1) return s;  
    String a = s.substring(0, N/2);  
    String b = s.substring(N/2, N);  
    return mystery(b) + mystery(a);  
}
```

Answer: `s` reversed

- f) (1 point) What does the following recursive function return, given two strings `s` and `t` of the same length?

```
public static String mystery(String s, String t) {  
    int N = s.length();  
    if (N <= 1) return s + t;  
    String a = mystery(s.substring(0, N/2), t.substring(0, N/2));  
    String b = mystery(s.substring(N/2, N), t.substring(N/2, N));  
    return a + b;  
}
```

Answer: Return a string with  $2i$ th character the  $i$ th character of `a` and  $2i + 1$ st character the  $i$ th character of `b` - for this answer we assume that we start counting the characters from 0.

- g) (1 point) What does the following code fragment print?

```
String string1 = "hello";  
String string2 = string1;  
string1 = "world";  
System.out.println(string2);
```

Answer: hello

h) (1 point) What does the following code fragment print?

```
String s = "Hello World";  
s.toUpperCase();  
s.substring(6, 11);  
System.out.println(s);
```

Answer: Hello World

i) (1 point) What does the following statement do, where c is of type char?

```
System.out.println((c >= 'a' && c <= 'z') ||  
                  (c >= 'A' && c <= 'Z')    );
```

Answer: True if c is a lower or upper case letter and false otherwise.

## Question 2 - Debugging [2 points]

a) (1 point) What happens when you execute the following code fragment?

```
String s = null;  
int length = s.length();
```

Answer: A NullPointerException is thrown.

b) (1 point) Why does program Bug1.java create a java.lang.NullPointerException when executed?

```
public class Bug1 {  
    private String s;  
    public void Bug1() { s = "hello"; }  
    public String toString() { return s; }  
    public static void main(String[] args) {  
        Bug1 x = new Bug1();  
        System.out.println(x);  
    }  
}
```

Answer: The programmer probably intended to make the no argument constructor set the string to hello. However, it has a return type (void) so it is an ordinary instance method instead of a constructor. It just happens to have the same name as the class.

### Question 3 - Immutable Classes [8 points]

Assume that the immutable `Vector` class represents finite sequences of `doubles`, which are interpreted as Cartesian coordinates, and that the API for the `Vector` class is as follows:

```
public class Vector
```

```
Vector(double[] a) - creates a vector with the given Cartesian coordinates
```

```
Vector plus(Vector b) - sum of this vector and b
```

```
Vector times(double t) - scalar product of this vector and t
```

Now complete the code for the `Vector` class below.

```
public class Vector {
    private int N;
    private double[] coords;
    public Vector(double[] a) {
        N = a.length;
        coords = new double[N];
        for (int i = 0; i < N; i++)
            coords[i] = a[i];
    }
    public Vector plus(Vector b) {
        double[] c = new double[N];
        for (int i = 0; i < N; i++)
            c[i] = coords[i] + b.coords[i];
        return new Vector(c);
    }
    public Vector times(double t) {
        double[] c = new double[N];
        for (int i = 0; i < N; i++)
            c[i] = t * coords[i];
        return new Vector(c);
    }
} }
```

## Question 4 - Performance [7 points]

- a) (3 points) Each of the three Java functions below returns a string of length  $N$  whose characters are all  $x$ . Determine the order of growth of the running time of each function. Recall that concatenating two strings in Java takes time proportional to the sum of their lengths.

A)

```
public static String method1(int N) {
    if (N == 0) return "";
    String temp = method1(N / 2);
    if (N % 2 == 0) return temp + temp;
    else          return temp + temp + "x";
}
```

Answer: Approximately a constant times  $N \log N$

B)

```
public static String method2(int N) {
    String s = "";
    for (int i = 0; i < N; i++)
        s = s + "x";
    return s;
}
```

Answer: Approximately a constant times  $(1+2+3+\dots+N)=N(N+1)/2$  which is approximately a constant times  $N^2$  if we ignore lower order terms.

C)

```
public static String method3(int N) {
    char[] temp = new char[N];
    for (int i = 0; i < N; i++)
        temp[i] = 'x';
    return new String(temp);
}
```

Answer: Approximately a constant times  $N$ .

- b) (1 point) What is the order of growth of the running time of the following function, which reverses a string  $s$  of length  $N$ ?

```
public static String reverse(String s) {
    int N = s.length();
    String reverse = "";
    for (int i = 0; i < N; i++)
        reverse = s.charAt(i) + reverse;
    return reverse;
}
```

Answer: Approximately a constant times  $(1 + 2 + 3 + \dots + N)$   
 $= N(N + 1)/2$  which is approximately a constant times  $N^2$  if we ignore lower order terms.

- c) [3 point ] Give the Java code for a linear algorithm for reversing a string.  
Answer:

```
public static String reverse(String s) {
    int N = s.length();
    char[] reverse = new char[N];
    for (int i = 0; i < N; i++)
        reverse[N-i-1] = s.charAt(i);
    return new String(reverse);
}
```