

# Methods and Final Review Lab

## [No Submission]



### CSE110: Programming Language I

You must not hard code any of the tasks, which means your code should work for any valid user input.

## Methods

---

### 1. [A,B,C,D should be written in a single java file]

- A. Write a method called **evenChecker** that takes an **integer** number as its argument and prints whether the number is even or odd **inside the method**.

| Sample Method Call            | Sample Output |
|-------------------------------|---------------|
| <code>evenChecker(10);</code> | Even!!        |
| <code>evenChecker(17);</code> | Odd!!         |

- B. Write a method called **isEven** that takes an **integer** number as an argument and **returns** boolean true if the number is even otherwise **returns** boolean false.

| Sample Method Call  | Sample Output |
|---|---------------|
| <code>boolean result = isEven(10);<br/>System.out.println( result );</code> | true          |
| <code>boolean result = isEven(17);<br/>System.out.println( result );</code> | false         |

- C. Write a method called **isPos** that takes an **integer** number as an argument and **returns** boolean true if the number is positive otherwise **returns** boolean false.

| Sample Method Call   | Sample Output |
|--|---------------|
| <code>boolean result = isPos(-5);<br/>System.out.println( result );</code> | false         |
| <code>boolean result = isPos(12);<br/>System.out.println( result );</code> | true          |

- D.** Write a method called **sequence()** that takes an **integer** in its parameter called **n**. Now, if **n** is **positive** then it prints all the **even** numbers from **0** to **n**, otherwise if **n** is **negative** it prints all the **odd** numbers from **n** to **-1**.

**Note:** **You must call** the methods from **CW-1B** and **CW-1C**, otherwise this task would be **considered invalid**.

| Sample Method Call         | Sample Output             | Explanation  |
|----------------------------|---------------------------|--|
| <code>sequence(10);</code> | <code>0 2 4 6 8 10</code> | Here, 10 is positive so 0,2,4,6,8,10 were printed. |
| <code>sequence(-7);</code> | <code>-7 -5 -3 -1</code>  | Here, -7 is negative so -7,-5,-3,-1 were printed.  |
| <code>sequence(7);</code>  | <code>0 2 4 6</code>      | Here, 7 is positive so 0,2,4,6 were printed        |
| <code>sequence(-8);</code> | <code>-7 -5 -3 -1</code>  | Here, -8 is negative so -7,-5,-3,-1 were printed.  |

## 2. [A,B,C should be written in a single java file]

- A.** Write a method called **circleArea** that takes an **integer** radius in its parameter and **returns** the **area** of the circle.

**Note:** area of a circle is  $\pi r^2$

| Sample Method Call  | Sample Output        |
|---|----------------------|
| <code>double area = circleArea(5);<br/>System.out.println(area);</code> | <code>78.5398</code> |

- B. Write a method called **sphereVolume** that takes an **integer** radius in its parameter and **returns** the **volume** of the sphere.

**Note:** volume of a sphere is  $\frac{4}{3}\pi r^3$

| Sample Method Call  | Sample Output |
|---|---------------|
| <pre>double volume = sphereVolume(5);<br/>System.out.println(volume);</pre> | 523.5987      |

- C. Write a method called **findSpace** that takes two values in its parameters one is an **integer** diameter and another one is a String. Using the given diameter, this method should calculate the Area of a circle or the Volume of a sphere depending on the value of the second parameter. Finally, it should print the result **inside the method**.

**Note:** **You must call** the method written in task **CW-2A & CW-2B**, otherwise this task would be **considered invalid**.

| Sample Method Call                 | Sample Output     |
|------------------------------------|-------------------|
| <pre>findSpace(10,"circle");</pre> | 78.5398           |
| <pre>findSpace(5,"sphere");</pre>  | 33.5103           |
| <pre>findSpace(10,"square");</pre> | "Wrong Parameter" |

3. Write a method called **isPrime** which takes an integer in its parameter to check whether a number is prime or not. If the number is prime then the method returns boolean **true** otherwise it returns boolean **false**.

| Sample Input  | Sample Output |
|---|---------------|
| <pre>boolean check = isPrime(7);<br/>System.out.println(check);</pre> | true          |

|  |       |
|--|-------|
| <code>boolean check = isPrime(15);<br/>System.out.println(check);</code> | false |
|--|-------|

4. Write a method called **isPerfect** which takes an integer in its parameter to check whether a number is perfect or not. If the number is perfect then the method returns boolean **true** otherwise it returns boolean **false**.

| Sample Input   | Sample Output |
|--|---------------|
| <code>boolean check = isPerfect(6);<br/>System.out.println(check);</code>  | true          |
| <code>boolean check = isPerfect(33);<br/>System.out.println(check);</code> | false         |

5. Write a method called **calcTax** that takes 2 arguments which are **your age** then **your salary**. The method must calculate and **return** the tax as per the following conditions:

- No tax if you are less than 18 years old.
- No tax if you get paid less than 10,000
- 7% tax if you get paid between 10K and 20K
- 14% tax if you get paid more than 20K

| Sample Method Call  | Output | Explanation  |
|---|--------|--|
| <code>double t = calcTax(16,20000);<br/>System.out.println(t);</code> | 0.0    | Here, the age is less than 18 so 0 tax.  |
| <code>double t = calcTax(20,18000);<br/>System.out.println(t);</code> | 1260.0 | Here, the age is greater than 18 and income is between 10K-20K so tax is 7% of 18000 = 1260. |

## Practice Problems

### Strings

---

#### **Task 1**

Trace the following code and write the outputs.

|  |
|--|
| public class Trace01{                    |
| public static void main(String[] args) { |
| String course = "";                      |
| int i = 2, j = 0, k = 18;                |
| course = "-->cse";                       |
| while (i< 5) {                           |
| k--;                                     |
| j = k;                                   |
| while (j > 12 ) {                        |
| if (j % 2 != 0) {                        |
| course += "<--";                         |
| course = course + i + (j / 2);           |
| } else {                                 |
| course += "-->";                         |
| course = course + (i % 2) + j;           |
| }  |
| System.out.println(course);              |
| if (j == 14) {                           |
| course = "-->cse";                       |
| }  |
| --j;                                     |
| }  |
| i++;                                     |
| }  |
| }  |
| }  |

## **Task 2**

Write a method **modifyStrings()** that takes in three given strings **S**, **S1**, and **S2** consisting of different numbers of characters respectively, the task is to modify the string **S** by **replacing** all the **substrings S1** with the **string S2** in the string **S** and printing the modified string **S**.

| Sample Input   | Sample Output | Explanation  |
|--|---------------|--|
| S = "abababa"<br>S1 = "aba"<br>S2 = "a"<br><br>modifyStrings(S, S1, S2); | aba           | Changing the substrings <b>S[0, 2]</b> (Referring to characters from the 0th index of <b>S</b> till the 2nd index of <b>S</b> and <b>S[4, 6]</b> (= <b>S1</b> ) to the string <b>S2</b> (= "a") modifies the string <b>S</b> to "aba". Therefore, print "aba". |
| S = "baddadda"<br>S1 = "dd"<br>S2 = "n"<br><br>modifyStrings(S, S1, S2); | banana        | Changing the substrings <b>S[2,3]</b> (Referring to characters from the 2nd and 3rd index of <b>S</b> ) and <b>S[5, 6]</b> (= <b>S1</b> ) to the string <b>S2</b> (= "n") modifies the string <b>S</b> to "banana". Therefore, print "banana".                 |

## Strings + Arrays

## **Task 3**

Given an array of email addresses, print the number of valid email addresses satisfying the following conditions.

- Each email contains an '@' character
- There is at least one character before and after '@' character and it has to start with letter
- There is a '.' character after the character(s) after '@' character
- There is at least one character after '.' character

| Sample Input   | Sample Output |
|--|---------------|
| email_list = {"abc@gmail.com", "!@cv.bd", "123cse@bracu.ac.bd"}        | 1             |
| email_list = {"cse110@gmail.com", "government@cv.", "eee@bracu.ac.bd"} | 2             |

## Strings + Methods

---

### **Task 4**

Write a method called `isHappyNumber` which takes an integer in its parameter to check whether a number is a happy number or not. If the number is a happy number then the method returns boolean `true` otherwise it returns boolean `false`. In number theory, a happy number is a number which eventually reaches 1 when replaced by the sum of the square of each digit. For instance, 13 is a happy number because  $1^2 + 3^2 = 10$  and  $1^2 + 0^2 = 1$ . On the other hand, 4 is not a happy number because the process continues in an infinite cycle without ever reaching 1. Unhappy number ends in a cycle of repeating numbers which contains 4 .

| Sample Input   | Sample Output      |
|--|--------------------|
| <code>boolean check = isHappyNumber(82)</code><br><code>System.out.println(check)</code> | <code>true</code>  |
| <code>boolean check = isHappyNumber(4)</code><br><code>System.out.println(check)</code>  | <code>false</code> |

### **Task 5**

Write a method called `toDecimal` which takes a binary number as a string in its parameter to convert the binary number to its decimal number and return the decimal value. After returning the decimal value, write another method called `toHex` which takes the converted decimal value in its parameter and calculates the hexadecimal value and then return the hex value.

| Sample Input   | Sample Output    |
|--|------------------|
| <code>int decimal = toDecimal("1010")</code><br><code>String hex = toHex(decimal)</code><br><code>System.out.println(hex)</code> | <code>"A"</code> |



## Arrays

---

### Task 6

Trace the following code and write the outputs.

|  |
|--|
| class Trace02 {                            |
| public static void main(String args[]) {   |
| int[] arr1 = {3, 1, 4, 1, 5, 9, 2};        |
| int[] arr2 = {10, 20, 30, 40, 50, 60, 70}; |
| int x = 0, y = 0;                          |
| while (x < arr1.length - 1) {              |
| arr2[x] = arr1[y] * (x + 1) - arr2[y];     |
| y = 1;                                     |
| while (y <= x) {                           |
| arr2[x] = arr2[x] + arr1[y] - y;           |
| y = y + 1;                                 |
| }  |
| System.out.println(arr2[x]);               |
| x = x + 1;                                 |
| }  |
| System.out.println(arr2[arr1.length - 1]); |
| }  |
| }  |

### Task 7

You are given an integer array. You need to identify all the **prime numbers** and **perfect numbers** within the array and print the **indices** along with these **numbers** from the original array.

| Sample Input  | Sample Output                                     |
|---|---|
| Sample Input:<br>int arr[] = {6, 13, 28, 17, 3, 9, 11, 23, 10, 29, 12, 7} | Prime Numbers:<br>1: 13<br>3: 17<br>4: 3<br>6: 11 |

|  |  |
|--|--|
|  | 7: 23<br>9: 29<br>11: 7<br><br>Perfect Numbers:<br>0: 6<br>2: 28 |
|--|--|

## **Task 8**

Trace the following code and write the outputs.

|   |
|---|
| public class tracing1 {                 |
| public static void main(String[] args){ |
| int i = 1;                              |
| int [] a = {5,6,7,8,9};                 |
| while (i <= 5){                         |
| int j = a[i%a.length];                  |
| while (j > 1){                          |
| System.out.print(j--);                  |
| if (j == 2)                             |
| break;                                  |
| }                                       |
| System.out.println("****");             |
| ++i;                                    |
| }                                       |
| double x = 7;                           |
| double y = 8;                           |
| double z = 9;                           |
| System.out.println(x < y    y > z);     |
| System.out.println(x < y && y > z);     |
| System.out.println(x < y);              |
| System.out.println(x + y < z);          |
| System.out.println((x + y)-6 < z);      |
| }                                       |
| }                                       |

## **Task 9**

Your professor expects only As, Bs, and Cs. In the following program, write a method called **getScores** that takes as input corresponding arrays **studentGrades** and **studentScores**. Write a method called **getScores** that assigns **index i** in **studentGrades** based on **index i** in **studentScores**. If a grade is **A**, assign **100**. If a grade is **B**, assign **90**. If a grade is **C**, assign **70**. If a grade is anything else, assign **0**.

| Sample Input   | Sample Output  |
|--|--|
| <pre>char[] studentGrades = new char[]{'A', 'A', 'A', 'B', 'C', 'U', 'Z'};<br/>int[] studentScores = new int[7];</pre> | <pre>Output expectation:<br/>100<br/>100<br/>100<br/>90<br/>70<br/>0<br/>0</pre> |

## Arrays + Methods

---

### **Task 10**

A. Write a method called **convertToCm()**, that takes as input a **type double** and **returns** the value converted from inches to centimeters.

**Hint:** There are 2.54 centimeters in an inch

| Sample Method Call  | Output   |
|---|----------|
| <pre>double t = convertToCm(16); System.out.println(t + " cm");</pre> | 40.64 cm |

B. Create an **array** of **type double** of length **5** called **cheetos\_inches**, that stores the length of each of the Cheetos **from the user**. Send the array of length in inches into a method called **findAvgCm()** that **returns** the average length of the Cheetos **in cm to 2 decimal places**. The method **findAvgCm()** uses **convertToCm()** to convert the length of each Cheetos **from inches to cm**.

**Note:** You must call the method written in [Method Task A], otherwise this task would be considered invalid.

| Sample Method Call   | Output                                |
|--|---------------------------------------|
| <pre>Sample array: double [] cheetos_inches = new double[]{10.0, 12.0, 14.0, 16.0, 18.0};  averageLength = findAvgCm(cheetos_inches);  System.out.println("The average Cheeto length is "+ averageLength + " cm");</pre> | The average Cheeto length is 35.56 cm |

## Arrays + Strings + Methods

---

### Task 11

A. Write a method called **isVowel** which takes a string in its parameter and counts all the vowels in the String. If any vowel exists in the string then the method returns the **count**.

| Sample Input                                | Sample Output                      |
|---|------------------------------------|
| The quick brown fox jumps over the lazy dog | Number of vowels in the string: 11 |

B. Write a method called **isConsonant** which takes a string in its parameter and counts all the consonants in the String. If any consonant exists in the string then the method returns the **count**.

| Sample Input                                | Sample Output                          |
|---|--|
| The quick brown fox jumps over the lazy dog | Number of consonants in the string: 24 |

C. Write a method called **vowel/consonantSum** which takes an array of strings in its parameter and returns the summation of the number of vowels/consonants.

**Note:** **You must call** the methods written in tasks A/B, otherwise this task will be **considered invalid**.

| Given Array  | Sample Output  |
|--|--|
| <pre>String [] names = {"Bob", "Alice", "Max", "Marry", "Rosy"}; System.out.println( "The total number of vowels in the array is:" + vowelSum(names)); System.out.println( "The total number of consonants in the array is:" + consonantSum(names));</pre> | <p>The total number of vowels in the array is: <b>7</b></p> <p>The total number of consonants in the array is: <b>13</b></p> |