
Lab 10

Final Review



CSE110: Programming Language I

Practice Problems

(No Submission)

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" S1 = "aba" S2 = "a" modifyStrings(S, S1, S2);	aba	Changing the substrings S[0, 2] (Referring to characters from the 0th index of S till the 2nd index of S and S[4, 6] (= S1) to the string S2 (= "a") modifies the string S to "aba". Therefore, print "aba".
S = "baddadda" S1 = "dd" S2 = "n" modifyStrings(S, S1, S2);	banana	Changing the substrings S[2,3] (Referring to characters from the 2nd and 3rd index of S) and S[5, 6] (= S1) to the string S2 (= "n") modifies the string S 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> <code>System.out.println(check)</code>	<code>true</code>
<code>boolean check = isHappyNumber(4)</code> <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> <code>String hex = toHex(decimal)</code> <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: int arr[] = {6, 13, 28, 17, 3, 9, 11, 23, 10, 29, 12, 7}	Prime Numbers: 1: 13 3: 17 4: 3 6: 11

	7: 23 9: 29 11: 7 Perfect Numbers: 0: 6 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'}; int[] studentScores = new int[7];</pre>	<pre>Output expectation: 100 100 100 90 70 0 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: 7</p> <p>The total number of consonants in the array is: 13</p>

Recursive Method Tracing

Task 12

Trace the following code to generate the outputs. Show the necessary trace table.

1	public class Tracel{
2	public static void main(String [] args){
3	int[] arr = {7, 2, 5};
4	for (int idx = 0; idx < arr.length; idx++){
5	System.out.println(findFibonacci(arr[idx]));
6	}
7	}
8	public static int findFibonacci(int n){
9	if (n < 2){
10	return n;
11	}
12	else{
13	return findFibonacci(n - 2) + findFibonacci(n - 1);
14	}
15	}
16	}

Task 13

Trace the following code to generate the outputs. Show the necessary trace table.

1	public class Trace2{
2	public static void main(String [] args){
3	methodA(5);
4	}
5	public static void methodA(int n){
6	if (n>=1){
7	methodA(n-1);
8	System.out.println(methodB(n));
9	}
10	}
11	public static int methodB(int a){
12	int b = (++a) + 6;
13	return b*a-a;
14	}
15	}

Task 14

Trace the following code to generate the outputs. Show the necessary trace table.

1	public class Trace3 {
2	public static void main(String[] args) {
3	int[] arr = {22, 7, 14, 33, 19};
4	int result = myMethod(arr, 0);
5	System.out.println(result);
6	}
7	public static int myMethod(int[] a, int idx) {
8	if (idx == a.length - 1) {
9	return a[idx];
10	}
11	int num = myMethod(a, idx + 1);
12	System.out.println(a[idx] + num);
13	if(a[idx] < num) {
14	return a[idx];
15	}
16	else{
17	return num;
18	}
19	}
20	}