**CSC272 Advanced Programming in Java**

**Assignment 2**

|  |  |
| --- | --- |
| **First Name** | Asa Jean |
| **Last Name** | Soriano |
| **ID#** | 041702547 |
| **Assignment Week#** | 2 |
| **Email Address** | asajeansoriano@gmail.com |

Contents

[How to submit your Assignment 1](#_Toc52035514)

[Problem 1 – *Prime Number and Prime Factors* 2](#_Toc52035515)

[Problem 2 – *Palindromes* 3](#_Toc52035516)

[Problem 3 – *Recursive Binary Search* 4](#_Toc52035517)

# How to submit your Assignment

After filling all the parts in this file, please follow the following steps.

1. Add your name and ID to the first page.
2. Save the file in the original format (Docx or Doc)

(Please **do not** convert to other file formats e.g. PDF, ZIP, RAR, …).

1. Rename the file as

*YOUR* ***First*** *Name - YOUR* ***Last*** *Name- YOUR student ID-* CSC272*.docx*

**Example:**

John – Smith - 234566435 – CSC272.docx

1. Upload the file and submit it (only using Blackboard)

Please do not hesitate to contact me, should you have any questions.

# Problem 1 – *Prime Number and Prime Factors*

Write a program that takes a whole number input from a user and determines whether it’s prime. If the number is not prime, display its unique prime factors. Remember that a prime number’s factors are only 1 and the prime number itself. Every number that’s not prime has a unique prime factorization. For example, consider the number 54. The prime factors of 54 are 2, 3, 3 and 3. When the values are multiplied together, the result is 54. For the number 54, the prime factors output should be 2 and 3. Use Sets as part of your solution. Please use one of the generic classes to implement this application.

|  |
| --- |
| Your code for this problem |
| import java.util.Scanner;  import java.util.Formatter;  import java.util.HashMap;  public class P1 {  //program that takes an integer input from user and determines if it is a prime number  //if the integer is not a prime, put the prime factors of that number in a HashMap  public static void main(String[] args) {  Scanner in = new Scanner(System.in);  System.out.println("Enter an integer: ");  int input = in.nextInt();  in.close();  HashMap<Integer, Integer> prime\_factors = new HashMap<Integer, Integer>();  int i = 2;  while (input > 1) {  if (input % i == 0) {  if (prime\_factors.containsKey(i)) {  prime\_factors.put(i, prime\_factors.get(i) + 1);  } else {  prime\_factors.put(i, 1);  }  input /= i;  } else {  i++;  }  }  System.out.println("The prime factors are: ");  //print all prime\_factors  for (Integer key : prime\_factors.keySet()) {  System.out.println(key + "^" + prime\_factors.get(key));  }  }    } |

Run the code and insert the result in the following box.

|  |
| --- |
| Sample Run Result |
|  |

# Problem 2 – *Palindromes*

A palindrome is a string that is spelled the same way forward and backward. Some examples of palindromes are “radar,” “able was i ere i saw elba” and (if spaces are ignored) “a man a plan a canal panama.” Write a recursive method testPalindrome that returns boolean value true if the string is a palindrome and false otherwise. The method should ignore spaces and punctuation in the string.

|  |
| --- |
| Your code for this problem |
| import java.util.Scanner;  public class P2 {  //program that writes a recursive function testPalindrome that returns true if string is a palindrome and false otherwise (ignore spaces and punctuation)  public static void main(String[] args) {  Scanner in = new Scanner(System.in);  System.out.println("Enter a string: ");  String input = in.nextLine();  in.close();  if (testPalindrome(input)) {  System.out.println("The string is a palindrome.");  } else {  System.out.println("The string is not a palindrome.");  }  }  public static boolean testPalindrome(String input) {  if (input.length() == 0) {  return true;  } else if (input.length() == 1) {  return true;  } else if (input.charAt(0) == input.charAt(input.length() - 1)) {  return testPalindrome(input.substring(1, input.length() - 1));  } else {  return false;  }  }  } |

Run the code and insert the result in the following box.

|  |
| --- |
| Sample Run Result |
|  |

# Problem 3 – *Recursive Binary Search*

Modify Fig.19.3 to use recursive method recursiveBinarySearch to perform a binary search of the array. The method should receive the search key, starting index and ending index as arguments. If the search key is found, return its index in the array. If the search key is not found, return –1.

|  |
| --- |
| Your code for this problem |
| import java.util.Scanner;  public class P3 {  //Recursive Binary Search  //recursiveBinarySearch performs binary search of an array  //input: array, target, start, end  //output: index of target in array or -1 if target is not in array  public static void main(String[] args) {  while(true) {  Scanner in = new Scanner(System.in);  int[] array = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  int start = 0;  int end = array.length - 1;  System.out.println("Array: ");  for (int i = 0; i < array.length; i++) {  System.out.print(array[i] + " ");  }  System.out.println();  System.out.println("Enter a target integer: ");  int target = Integer.parseInt(in.nextLine());  System.out.println("Target: " + target);  System.out.println("Start: " + start);  System.out.println("End: " + end);  int index = recursiveBinarySearch(array, target, start, end);  if (index == -1) {  System.out.println("Target not found.");  } else {  System.out.println("Target found at index " + index);  }  System.out.println("~~~~~~~~~");  //in.close();  }  }  public static int recursiveBinarySearch(int[] array, int target, int start, int end) {  if (start > end) {  return -1;  } else {  int mid = (start + end) / 2;  if (array[mid] == target) {  return mid;  } else if (array[mid] > target) {  return recursiveBinarySearch(array, target, start, mid - 1);  } else {  return recursiveBinarySearch(array, target, mid + 1, end);  }  }  }  } |

Run the code and insert the result in the following box.

|  |
| --- |
| Sample Run Result |
|  |

**The end**