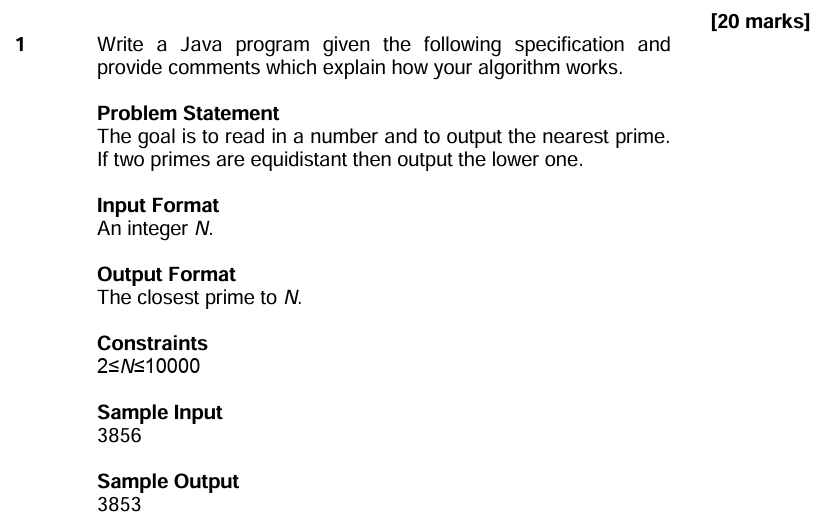
# Question 1



**import** java.util.Scanner;

**public** **class** Q1 {

**public** **static** **void** main (String args[]) {

Scanner sc = **new** Scanner(System.***in***);

**int** inputNum = sc.nextInt();

sc.close();

**int** nearestPrime = *findNearestPrime*(inputNum);

System.***out***.println(nearestPrime);

}

// Find nearest Prime

**public** **static** **int** findNearestPrime(**int** input) {

**int** lowerPrime = input;

**int** upperPrime = input;

**while**(!*isPrime*(lowerPrime)) lowerPrime--;

**while**(!*isPrime*(upperPrime)) upperPrime++;

**if**(Math.*abs*(lowerPrime - input) > Math.*abs*(lowerPrime -input)) {

**return** upperPrime;

}

// If two primes are equidistant then output the lower one.

**else** {

**return** lowerPrime;

}

}

//Check whether the number is Prime

**public** **static** **boolean** isPrime(**int** number) {

**if** (number <= 1) **return** **false**;

**for**(**int** i = 2; i < number; i++) {

**if**(number % i == 0) **return** **false**;

}

**return** **true**;

}

}

# Question 2

文本, 信件

描述已自动生成

**import** java.util.LinkedList;

**import** java.util.Collections;

**import** java.util.Scanner;

**public** **class** Q2 {

**public** **static** **void** main (String args[]) {

Scanner sc = **new** Scanner(System.***in***);

LinkedList<String> list = **new** LinkedList<String>();

// input the number of words to be sorted

**int** N = Integer.*parseInt*(sc.nextLine());

// input a line with N words, and put it in List

**for**(**int** i = 0; i < N; i++) {

String inputLine = sc.nextLine();

list.add(inputLine);

}

sc.close();

// Sort in reverse alphabetical order.

Collections.*sort*(list, Collections.*reverseOrder*());

// Print out the sorted words

**for**(String s : list) {

System.***out***.println(s);

}

}

}

# Question 3

文本, 信件

描述已自动生成

## Answer 1 – Interface

**import** java.util.Stack;

**import** java.util.Scanner;

**public** **class** Q3\_Interface {

**public** **static** **void** main (String args[]) {

Scanner sc = **new** Scanner (System.***in***);

String sentence = sc.nextLine();

sc.close();

**if**(*isPalindrome*(sentence)) {

System.***out***.println("TRUE");

}

**else** {

System.***out***.println("FALSE");

}

}

// Check whether the String is Palindrome

**public** **static** **boolean** isPalindrome(String input) {

Stack<Character> s = **new** Stack<Character>();

// Convert String to char Array, all lower case.

**char** []charArray = input.toLowerCase().toCharArray();

// Input all char from first to last

**for** (**char** c: charArray) s.push(c);

// Check char from last to first

**for** (**char** c: charArray) {

**if**(c != s.pop()) **return** **false**;

}

**return** **true**;

}

}

## Answer 2 – Full Stack Class

**import** java.util.Scanner;

**public** **class** Q3\_FullStackClass {

**public** **static** **void** main (String args[]) {

Scanner sc = **new** Scanner (System.***in***);

String sentence = sc.nextLine();

sc.close();

**if**(*isPalindrome*(sentence)) {

System.***out***.println("TRUE");

}

**else** {

System.***out***.println("FALSE");

}

}

// Check whether the String is Palindrome

**public** **static** **boolean** isPalindrome(String input) {

FullStack s = **new** FullStack(100);

// Convert String to char Array, all lower case.

**char** []charArray = input.toLowerCase().toCharArray();

// Input all char from first to last

**for** (**char** c: charArray) s.push(c);

// Check char from last to first

**for** (**char** c: charArray) {

**if**(c != s.pop()) **return** **false**;

}

**return** **true**;

}

}

**class** FullStack{

**private** **int** maxSize; // size of stack array

**private** **char**[] stackArray;

**private** **int** top; // top of stack

**public** FullStack(**int** s) { // constructor

maxSize = s; // set array size

stackArray = **new** **char**[maxSize]; // create array

top = -1; // no items yet

}

**public** **void** push(**char** j) { // nput item on top of stack

top++;

stackArray[top] = j; // increment top, insert item

}

**public** **char** pop() { // take item from top of stack

**return** stackArray[top--]; //access item, decrement top

}

}

# Question 4

## Question a

文本

描述已自动生成

**The program runs main function first, it will call method(14)**

**1) method(14).**

14 % 7 = 0, 0 != 3 => skip if statement

print out **“hello”, then change line.**

**return method((14 % 5) + 3) - 2 = method(7) - 2**

**2) method(7).**

7 % 7 = 0, 0 != 3 => skip if statement

print out **“hello”, then change line.**

**return method((7 % 5) + 3) - 2 = method(5) - 2**

**3) method(5).**

5 % 7 = 5, 5 != 3 => skip if statement

print out **“hello”, then change line.**

**return method((5 % 5) + 3) - 2 = method(3) - 2**

**4) method(3).**

3 % 7 = 5, 3 == 3 =>run if statement

**return 5**

* **method(3) = 5**

**5) Calling method(5)**

**method(5) = method(3) – 2 = 5 – 2 = 3**

**get 3**

**6) Calling method(7)**

**method(7) = method(5) – 2 = 3 – 2 = 1**

**get 1**

**7) Calling method(14)**

**method(14) = method(7) – 2 = 1 – 2 = -1**

**get -1**

**Therefore, the Java Program outputs**

**`**

**hello**

**hello**

**hello**

**-1**

**`**

**when it runs.**

## Question b

文本

描述已自动生成

**The program will print out the equation**

**(((4|6)|(5&3))<<5)**

Step 1: 4 & 6

|  |  |  |
| --- | --- | --- |
| **(4)10** | **=(00000100)2** |  |
| **(7)10** | **=(00000110)2** | **|** |
|  | **`(00000110)2** | **= (6)10** |

Step 2: 5 & 3

|  |  |  |
| --- | --- | --- |
| **(5)10** | **=(00000101)2** |  |
| **(3)10** | **=(00000011)2** | **&** |
|  | **`(00000001)2** | **= (1)10** |

Step 3: 6 | 1

|  |  |  |
| --- | --- | --- |
| **(6)10** | **=(00000110)2** |  |
| **(1)10** | **=(00000001)2** | **&** |
|  | **`(00000111)2** | **= (7)10** |

Step 4: 7 << 5

**(00000111)2 << 5 = (11100000)2 = (224)10**

**Therefore, the Java Program outputs 224 when it runs.**