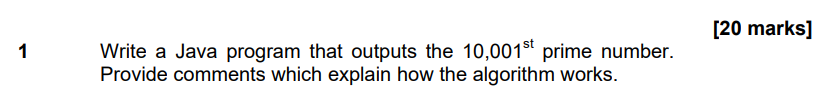
# Question 1



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

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

**int** count = 0;

**int** number = 2;

// Find the nTH Prime number

**while**(count < 10001) {

**if**(*isPrime*(number)) count++;

number++;

}

/\* When go out the loop, return the previous number, that is the answer \*/

number--;

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

}

//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

描述已自动生成

**import** java.util.Stack;

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

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

System.***out***.println(*isPalindrome*("acca"));

System.***out***.println(*isPalindrome*("abc"));

}

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

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

//Convert String into char Array

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

//Push each char into the Stack

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

//Pop each char out of the Stack and compare

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

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

}

**return** **true**;

}

}

# Question 3

文本

描述已自动生成

**class** Node {

**int** data;

Node next;

Node prev;

**public** Node(**int** data) {

**this**.data = data;

**this**.next = **null**;

**this**.prev = **null**;

}

}

**class** SingleEndedDoublyLinkedList {

Node head;

/\* Method to delete every third link in a single-ended doubly-linked list \*/

**public** **void** deleteEveryThirdLink() {

Node current = head;

**int** count = 0;

**while** (current != **null**) {

count++;

// Check if the current link is the third one

**if** (count % 3 == 0) {

Node prevNode = current.prev;

Node nextNode = current.next;

// Update links to skip the current node

**if** (prevNode != **null**) {

prevNode.next = nextNode;

// If current node is the head, update head

} **else** {

head = nextNode;

}

**if** (nextNode != **null**) {

nextNode.prev = prevNode;

}

// Move to the next node (skip the deleted one)

current = nextNode;

} **else** {

// Move to the next node

current = current.next;

}

}

}

}

# Question 4

图形用户界面, 文本

描述已自动生成

**import** java.util.Scanner;

**import** java.util.Comparator;

**import** java.util.LinkedList;

**import** java.util.Collections;

**public** **class** Q4 {

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

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

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

//Input the String, ends by empty String

**while**(**true**) {

String inputLine = sc.nextLine();

**if**(inputLine.isEmpty()) {

sc.close();

**break**;

}

list.add(inputLine);

}

//The sort method is in O(nlogn) time

Collections.*sort*(list, **new** Comparator<String>() {

@Override

**public** **int** compare(String o1, String o2) {

// 1 - short String has priority

**if**(o1.length() != o2.length()) {

**return** o1.length()-o2.length();

}

// 2 - Alphabetically

**else** {

**return** o1.compareTo(o2);

}

}

});

//Print out the sorted String

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

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

}

}

}

# Question 5

文本

描述已自动生成

**import** java.util.Set;

**import** java.util.TreeSet;

**public** **class** Q5 {

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

//Monte Carlo Simulation

**int** N = 1000000;

**int** count = 0;

MONTECARLO:

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

/\* Create a lottery to store 6 numbers drawn from 1 to 45

drawn from 1 to 45, and sort them! \*/

Set<Integer> lottery = **new** TreeSet<>();

**while**(lottery.size() < 6) {

**int** draw = (**int**) (45 \* Math.*random*()) + 1;

lottery.add(draw);

}

//Convert TreeSet to Array

Integer[] lotteryArray = lottery.toArray(

**new** Integer[lottery.size()]);

//Check whether lottery has consecutive pairs of numbers

**for**(**int** j = 1; j < lotteryArray.length; j++) {

**if**(lotteryArray[j] == lotteryArray[j-1] + 1 ) {

// If has consecutive pairs of numbers, go to next loop

**continue** MONTECARLO;

}

}

// If does not have consecutive pairs of numbers, count it.

count++;

}

// Print out the probability, retain 2 digits

**double** probability = (**double**)(100 \* count) / (**double**)N;

System.***out***.printf("Probability of Lottery: %.2f", probability);

}

}