# Question 1

文本, 信件

描述已自动生成

**import** java.util.Scanner;

**import** java.util.Queue;

**import** java.util.PriorityQueue;

**import** java.util.Comparator;

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

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

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

Queue<Integer> pq = **new** PriorityQueue<Integer>(

**new** Comparator<Integer>() {

@Override

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

**return** *getCollatzSteps*(o2) - *getCollatzSteps*(o1);

}

});

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

String inputLine = sc.nextLine();

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

sc.close();

**break**;

}

**int** inputNum = Integer.*parseInt*(inputLine);

pq.add(inputNum); // O(n\*logn) n: number of input

}

**while** (!pq.isEmpty()) {

System.***out***.println(pq.poll());

}

}

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

**int** steps = 0;

**while**(input>1) { //

**if**(input % 2 == 0) {

input = input / 2;

}

**else** {

input = input \* 3 + 1;

}

steps++;

}

**return** steps;

}

}

/\*

Big-O Complexity is time complexity, it is a concept that describes

the speed of an algorithm according to the size of input.

The Big-O Complexity in my program is O(n\*logn)

\*/

# Question 2

图形用户界面, 文本, 应用程序

描述已自动生成

**图片包含 徽标

描述已自动生成**

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

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

**int** totalSimulations = 1000000; // Number of simulations

**int** countBThirdPlace = 0; // Counter for Horse B finishing third

**for** (**int** i = 0; i < totalSimulations; i++) { // A[0,53) B[53,79) C[79,93) D 93,100]

**double** randomValue = Math.*random*() \* 100; // Generate a random number between 0 and 100

// Determine First Place Horse

**char** firstPlace = (randomValue < 53) ? 'A' : (randomValue < 79) ? 'B' :

(randomValue < 93) ? 'C' : 'D';

// Determine Second Place Horse (exclude First Place)

randomValue = Math.*random*() \* (100 - *getProbability*(firstPlace));

**char** secondPlace;

**if** (firstPlace == 'A') { // B[0,26) C[26,40) D[40,47]

secondPlace = (randomValue < 26) ? 'B' : (randomValue < 40) ? 'C' : 'D';

} **else** **if** (firstPlace == 'B') { // A[0,53) C[53,67) D[67,74]

secondPlace = (randomValue < 53) ? 'A' : (randomValue < 67) ? 'C' : 'D';

} **else** **if** (firstPlace == 'C') { // A[0,53) B[53,79) D[79,86]

secondPlace = (randomValue < 53) ? 'A' : (randomValue < 79) ? 'B' : 'D';

} **else** { // A[0,53) B[53,79) C[79,93]

secondPlace = (randomValue < 53) ? 'A' : (randomValue < 79) ? 'B' : 'C';

}

// Determine Third Place Horse (exclude First Place and Second Place)

randomValue = Math.*random*() \* (100 - *getProbability*(firstPlace) - *getProbability*(secondPlace));

**char** thirdPlace;

**if** ((firstPlace == 'A' && secondPlace == 'B') || (firstPlace == 'B' &&

secondPlace == 'A')) {

thirdPlace = (randomValue < 14) ? 'C' : 'D'; // C[0,14) D[14,21]

} **else** **if** ((firstPlace == 'A' && secondPlace == 'C') || (firstPlace == 'C' &&

secondPlace == 'A')) {

thirdPlace = (randomValue < 26) ? 'B' : 'D'; // B[0,26) D[26,33]

} **else** **if** ((firstPlace == 'A' && secondPlace == 'D') || (firstPlace == 'D' &&

secondPlace == 'A')) {

thirdPlace = (randomValue < 26) ? 'B' : 'C'; // B[0,26) C[26,40]

} **else** **if** ((firstPlace == 'B' && secondPlace == 'C') || (firstPlace == 'C' &&

secondPlace == 'B')) {

thirdPlace = (randomValue < 53) ? 'A' : 'D'; // A[0,53) D[53,60]

} **else** **if** ((firstPlace == 'B' && secondPlace == 'D') || (firstPlace == 'D' &&

secondPlace == 'B')) {

thirdPlace = (randomValue < 53) ? 'A' : 'C'; // A[0,53) C[53,67]

}**else** {

thirdPlace = (randomValue < 53) ? 'A' : 'B'; // A[0,53) B[53,79]

}

// If Horse B finishes third, increment the counter

**if** (thirdPlace == 'B') {

countBThirdPlace++;

}

}

// Calculate and print the estimated probability

**double** estimatedProbability = (**double**) 100 \* countBThirdPlace / totalSimulations;

System.***out***.printf("Estimated probability that Horse B finishes third: %.2f %%", estimatedProbability);

}

// Helper method to get probability based on horse

**private** **static** **double** getProbability(**char** horse) {

**switch** (horse) {

**case** 'A': **return** 53;

**case** 'B': **return** 26;

**case** 'C': **return** 14;

**case** 'D': **return** 7;

**default**: **return** 0;

}

}

}

# Question 3

图形用户界面, 文本, 应用程序

描述已自动生成

## Answer 1 – Interface

**import** java.util.Queue;

**import** java.util.LinkedList;

**import** java.util.Scanner;

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

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

Queue<Integer> q = **new** LinkedList<Integer>();

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

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

String input = sc.nextLine();

**if**(input.isEmpty()) {

sc.close();

**break**;

}

**if**(input.split(" ")[0].toUpperCase().equals("INSERT")){

String insertNumStr = input.split(" ")[1];

**int** insertNum = Integer.*parseInt*(insertNumStr);

q.add(insertNum);

}

**if**(input.toUpperCase().equals("REMOVE")) {

/\* If a remove command is issued for an empty queue

then nothing should happen. \*/

**if**(!q.isEmpty()) q.remove();

}

}

System.***out***.println(q.peek());

}

}

## Answer 2 – Full Queue Class

**import** java.util.Scanner;

**public** **class** Q3\_FullQueueClass {

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

FullQueue q = **new** FullQueue(100);

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

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

String input = sc.nextLine();

**if**(input.isEmpty()) {

sc.close();

**break**;

}

**if**(input.split(" ")[0].toUpperCase().equals("INSERT")){

String insertNumStr = input.split(" ")[1];

**int** insertNum = Integer.*parseInt*(insertNumStr);

q.insert(insertNum);

}

**if**(input.toUpperCase().equals("REMOVE")) {

/\* If a remove command is issued for an empty queue

then nothing should happen. \*/

**if**(!q.isEmpty()) q.remove();

}

}

System.***out***.println(q.remove());

}

}

**class** FullQueue{

**private** **int** maxSize;

**private** **long**[] queArray;

**private** **int** front;

**private** **int** rear;

**private** **int** nItems;

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

maxSize = s;

queArray = **new** **long**[maxSize];

front = 0;

rear = -1;

nItems = 0;

}

**public** **boolean** insert(**long** j) { // put item at rear of queue

// deal with wraparound

**if**(rear == maxSize - 1) {

rear = -1;

}

rear++;

queArray[rear] = j; // increment rear and insert

nItems++; // one more item

**return** **true**; //successfully inserted

}

**public** **long** remove() { // take item from front of queue

**if**(isEmpty()) **return** (Long) **null**; //don’t remove if empty

**long** temp = queArray[front];// get value and incr front

front++;

// deal with wraparound

**if**(front == maxSize) front = 0;

nItems--; // one less item

**return** temp;

}

**public** **boolean** isEmpty() { // true if queue is empty

**return** (nItems==0);

}

}

# Question 4

## Question a

文本

描述已自动生成

**The program runs main function first, it will call function(“Launch”)**

**1) function(“Launch”).**

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

length = 6, 6 % 7 != 0 => skip if statement

**return(function(“LaunchReturn”)+"Terminated")**

**2) function(“LaunchReturn”).**

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

length = 12, 12 % 7 != 0 => skip if statement

**return(function(“LaunchReturnReturn”)+"Terminated")**

**3) function(“LaunchReturnReturn”).**

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

length = 18, 18 % 7 != 0 => skip if statement

**return(function(“LaunchReturnReturnReturn”)+"Terminated")**

**4) function(“LaunchReturnReturnReturn”).**

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

length = 24, 24 % 7 != 0 => skip if statement

**return(function(“LaunchReturnReturnReturnReturn”)+"Terminated")**

**5) function(“LaunchReturnReturnReturnReturn”).**

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

length = 30, 30 % 7 != 0 => skip if statement

**return(function(“LaunchReturnReturnReturnReturnReturn”)+"Terminated")**

**6) function(“LaunchReturnReturnReturnReturnReturn”).**

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

length = 36, 36 % 7 != 0 => skip if statement

**return(function(“LaunchReturnReturnReturnReturnReturnReturn”)+"Terminated")**

**7) function(“LaunchReturnReturnReturnReturnReturnReturn”).**

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

length = 42, 42 % 7 == 0 => run if statement

**return(“Exit”)**

**8) Calling function(“LaunchReturnReturnReturnReturnReturnReturn”)+"Terminated"**

**Get “ExitTerminated”**

**9) Calling function(“LaunchReturnReturnReturnReturnReturn”)+"Terminated"**

**Get “ExitTerminatedTerminated”**

**10) Calling function(“LaunchReturnReturnReturnReturn”)+"Terminated"**

**Get “ExitTerminatedTerminatedTerminated”**

**11) Calling function(“LaunchReturnReturnReturn”)+"Terminated"**

**Get “ExitTerminatedTerminatedTerminatedTerminated”**

**12) Calling function(“LaunchReturnReturn”)+"Terminated"**

**Get “ExitTerminatedTerminatedTerminatedTerminatedTerminated”**

**13) Calling function(“LaunchReturn”)+"Terminated"**

**Get “ExitTerminatedTerminatedTerminatedTerminatedTerminatedTerminated”**

Finally, print out **“ExitTerminatedTerminatedTerminatedTerminatedTerminatedTerminated”, then change line.**

**Therefore, the Java Program outputs**

**`**

**Evaluating**

**Evaluating**

**Evaluating**

**Evaluating**

**Evaluating**

**Evaluating**

**Evaluating**

**ExitTerminatedTerminatedTerminatedTerminatedTerminatedTerminated**

**`**

**when it runs.**

## Question b

图形用户界面, 文本, 应用程序

中度可信度描述已自动生成

**The program will print out the equation**

**(((11&19)|5)<<3)**

Step 1: 11 & 19

|  |  |  |
| --- | --- | --- |
| **(11)10** | **=(00001011)2** |  |
| **(19)10** | **=(00010011)2** | **&** |
|  | **`(00000011)2** | **= (3)10** |

Step 2: 3 | 5

|  |  |  |
| --- | --- | --- |
| **(3)10** | **=(00000011)2** |  |
| **(5)10** | **=(00000101)2** | **|** |
|  | **`(00000111)2** | **= (7)10** |

Step 3: 7 << 3

**(00000111)2 << 3 = (00111000)2 = (56)10**

**Therefore, the Java Program outputs 56 when it runs.**

## \*Question c

文本

描述已自动生成

**public** **class** Q4\_C {

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

SinglyLinkedList list = **new** SinglyLinkedList();

// Adding nodes to the list

Node current = list.head;

**for**(**int** i=1; i<=10; i++) {

Node newNode = **new** Node(i);

**if** (list.head == **null**) {

list.head = newNode;

} **else** {

Node temp = list.head;

**while** (temp.next != **null**) {

temp = temp.next;

}

temp.next = newNode;

}

}

System.***out***.println("Original List:");

list.printList();

// Delete every third node

list.deleteEveryThirdNode();

System.***out***.println("List after deleting every third node:");

list.printList();

}

}

**class** Node {

**public** **int** data;

**public** Node next;

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

**this**.data = data;

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

}

}

**class** SinglyLinkedList {

Node head;

**public** SinglyLinkedList() {

head = **null**;

}

// Method to delete every third node in the linked list

**public** **void** deleteEveryThirdNode() {

**if** (head == **null** || head.next == **null** || head.next.next == **null**) {

// If the list is empty or has fewer than three nodes, no deletion is needed

**return**;

}

Node currentNode = head;

Node previousNode = **null**;

**int** count = 1;

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

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

// Delete the current node

previousNode.next = currentNode.next;

count = 1; // Reset the count after deletion

} **else** {

count++;

}

previousNode = currentNode;

currentNode = currentNode.next;

}

}

// Method to print the linked list

**public** **void** printList() {

Node currentNode = head;

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

System.***out***.print(currentNode.data + " ");

currentNode = currentNode.next;

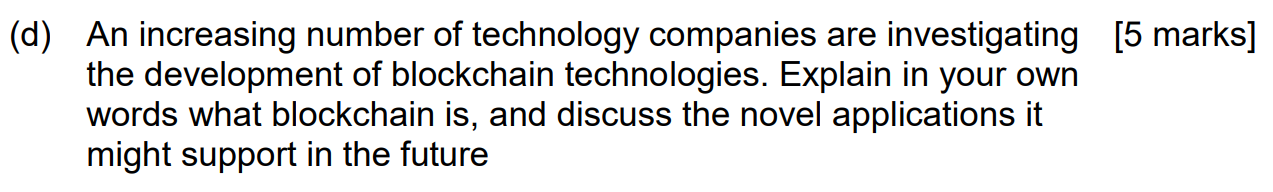
}

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

}

}

## \*Question d



**Blockchain technology is a digital ledger technology. Each block contains a set of transactions, and once a block is completed, it is added to the chain in linear chronological order. Blockchain is distributed across a network of computers, making it highly transparent and secure.**

**Novel Applications that BlockChain may support in the future:**

1. **Smart Contracts:**
2. **Supply Chain Management:**
3. **Healthcare Record Management:**
4. **Voting Systems:**
5. **Decentralized Finance (DeFi):**
6. **Identity Verification:**
7. **Energy Trading:**
8. **Intellectual Property and Royalties:**
9. **Education and Academic Credentials:**
10. **Internet of Things (IoT):**