Part I. Multiple Choice Questions

For each multiple choice question circle **only one** answer.

1. (1 mark) A constructor in a Java class can invoke both public and private methods from that class.

   [(A) True] (B) False

2. (1 mark) A Java interface may contain a constructor and/or instance variables.

   (A) True [(B) False]

3. (1 mark) A subclass A of a Java class B can be the parent of one or more other classes.

   [(A) True] (B) False

4. (1 mark) All Java classes either directly or indirectly extend the class `Object`.

   [(A) True] (B) False

5. (2 mark) Consider the following code fragment

```
String toto = new String("Allo");
String titi = new String("Allo");
if (toto == titi) System.out.println("equal");
else System.out.println("unequal");
```

   What is the output produced by the above code fragment?

   (A) `"equal"` [(B) `"unequal"`] (C) Nothing. The third line of the code causes a runtime error.

6. (2 marks) Consider the following Java interface for the stack ADT

```
public interface StackADT<T> {
   public void push(T element);
   public T pop();
}
```

   and the following Java code fragment:

```
StackADT<String> s = new StackADT<String>();
s.push("hello");
s.push("hi");
System.out.println(s.pop());
```

   What is printed by this above code?

   (A) `"hi"` (B) `"hello"` [(C) Nothing. The above code has compilation errors.]

7. (3 marks) Consider the following code fragment

```
String s;
String[] arr = new String[5];
try {
   s = arr[0];
   for (int i = 0; i < s.length(); ++i) s = s + arr[i];
}
catch (ArrayIndexOutOfBoundsException e) {
   System.out.println("Invalid index");
}
catch (NullPointerException e) {System.out.println("Null pointer");}
catch (Exception e) {System.out.println("Exception");}
```

   What is printed when the above code is executed?

   (A) "Invalid index" [(B) "Null pointer"] (C) "Exception" (D) Nothing
   (E) "Invalid index", "Null pointer", and "Exception"

8. (3 marks) Let class A be a subclass or child class of class B. A() and B() are constructors for these classes. Consider the following code fragment

```
A refA;
B refB;
refA = new B(); //Line 1
refB = new A(); // Line 2
```

Which line(s) generate compilation error(s)? (Line numbers are indicated in the comments).

(A) Line 1 (B) Line 2 (C) Lines 1 and 2 (D) None

9. (2 marks) Let class A be a subclass of class B. A() and B() are constructors for these classes. Let `private int a = 0;` be an instance variable of class A and `private int b = 1;` be an instance variable of class B. Consider the following code fragment

```
A refA = new A();
B refB = new B();
refA.a = refB.b;
```

Which of the following statements is true?

(A) After executing the code `refA.a` has value 1

(B) The code has compilation and/or runtime errors

10. (2 marks) Let class A be a subclass of class B. A() and B() are constructors for these classes. Let `m()` be a public method defined in class A and let `m()` be a public method defined in class B as well (so method `m()` in class A overrides method `m()` in class B). Consider the following code fragment

```
A refA = new A();
B refB = (B) refA;
refB.m();
```

Which of the following statement is correct?

(A) The above code has compilation and/or runtime errors.

(B) There are no errors and in the last statement, method `m()` from class A is invoked.

(C) There are no errors and in the last statement, method `m()` from class B is invoked.

11. (2 marks) Consider the following Java code

```
public class A{
   private static int x = 0;
   private static void change(int x) {
       x = 3;
   }
   public static void main (String[] args) {
       change(3);
       System.out.println(x);
   }
}
```

Which value is printed when method `main` is executed?

(A) 0 (B) 3

12. (2 marks) Consider a stack `s` and a queue `q` storing integer values. For the following code fragment

```
int i;
for (i = 0; i < 6; i = i+1) s.push(i);
for (i = 0; i < 3; i = i + 1) q.enqueue(s.pop());
for (i = 0; i < 3; i = i + 1) s.push(q.dequeue());
System.out.println(s.pop());
```

What value is printed?

(A) 2 (B) 3 (C) 4 (D) 5

13. (2 marks) Consider the following code fragment

```
public class A {
   public void m1() { System.out.println("m1"); }
   public void m2() { System.out.println("m2"); }
   public static void main (String[] args) {
       m1(); // Line 1
       m2(); // Line 2
   }
}
```

Which lines have compilation errors?

(A) Line 1 (B) Line 2 [(C) Line 1 and line 2] (D) None

14. (3 marks) Consider the following Java class

```
public class B {
   public static int x;
   public int y;
   public B() {
       x = 0;
       y = -1;
   }
   public void inc() {
       ++x;
       ++y;
   }
}
```

and the following code fragment

```
B ref1 = new B();
B ref2 = new B();
ref1.inc();
ref2.inc();
```

Which values do `ref2.x` and `ref2.y` have after executing this code?

[(A) `x = 2, y = 0`] (B) `x = 2, y = 1` (C) `x = 1, y = 0` (D) `x = 0, y = -1`

15. (2 marks) Consider the following two Java classes.

```
public class B {
   private void m1() { System.out.println("m1"); }
   public void m2() { System.out.println("m2"); }
   public B() {}
}
public class A extends B {
   public A() {
       B varB = new B();
       varB.m1(); // Line 1
       varB.m2(); // Line 2
   }
}
```

Which line(s) will cause compilation error(s)?

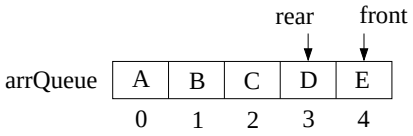
[(A) Line 1] (B) Line 2 (C) Lines 1 and 2 (D) None

16. (2 marks) What is the output produced by the following program?

```
public class parameters {
   public static void m(int a, int b) {
       int temp;
       temp = a;
       a = b;
       b = temp;
   }
   public static void main (String[] args) {
       int a = 1;
       int b = 2;
       m(a,b);
       System.out.println(a + ", " + b);
   }
}
```

(A) 1, 1 (B) 2, 2 **(C) 1, 2** (D) 2, 1

17. (2 marks) Consider the following queue implemented using a circular array.



Which code fragment correctly implements the `dequeue` operation on this particular queue? The name of the array storing the data items in the queue is `arrQueue`.

(A) `r = arrQueue[front]; front = front + 1; count--; return r;`
(B) `r = arrQueue[rear]; rear = rear - 1; count--; return r;`
(C) `r = arrQueue[front]; front = (front + 1) % arrQueue.length; count--; return r;`
(D) `r = arrQueue[front]; rear = (rear + 1) % arrQueue.length; count--; return r;`
(E) `r = arrQueue[front]; front = (front - 1) % arrQueue.length; count--; return n;`

Part II. Written Answers

18. (7 marks) Consider the following Java code.

```
public class A {
  private static int[] a;
  private static void m(int x) throws Exception1, Exception2 {
    try {
      for (int i = 0; i <= x; ++i) if (x > 1) a[i] = i; else a[i] = 2 * a[i] + 1;
      a[1] = 10;
    }
    catch (Exception e) {
      a[0] = x;
      throw new Exception1();
    }
    throw new Exception2();
  }
  public static void main(String[] args) {
    a = new int[3];
    try {
      m(3);
    }
    catch (Exception1 e) {a[1] = 20;}
    catch (Exception2 e) {a[1] = -20;}
  }
}
```

`Exception1` and `Exception2` are not parent/child classes of each other. What values are stored in array `a` after the above code is executed? Your answer must be in the form a[0] = ..., a[1] = ...

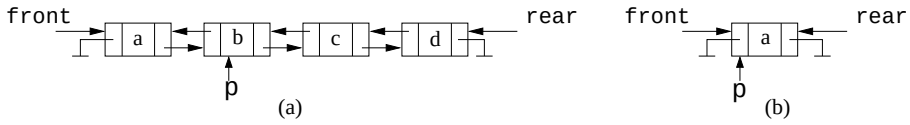
`a[0] = 3, a[1] = 20, a[2] = 2`

19. (8 marks) The following code is intended to remove a node `p` from a doubly linked list:

```
next = p.getNext();
next.setPrevious(p.getPrevious());
prev = p.getPrevious();
p.setNext(prev.getNext());
if (p == front) front = next;
if (p == rear) rear = prev;
```

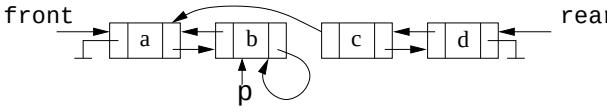
where `getNext`, `getPrevious`, `setNext`, and `setPrevious` are a getter and setter methods from the class representing the nodes of the linked list to access and set the next node and previous node in the list.

Consider the following two doubly linked lists



(4 marks) Does the above code correctly remove node `p` from list (a)? If the code can be executed without crashing, draw the doubly linked list resulting after it is executed on the above list (a); otherwise explain why the code would crash (i.e. an exception would be thrown).

No, the above code does not correctly remove node `p` from the list. After executing the code the list would look like this:



(4 marks) Does the above code correctly remove node `p` from list (b)? If the code can be executed without crashing, draw the doubly linked list resulting after it is executed on the above list (b); otherwise explain why the code would crash (i.e. an exception would be thrown).

No, the above code does not correctly remove node p. In the first line of the code we set `next = p.getNext() = null`. Hence the second line causes a null pointer exception.

For each one of the following three questions you need to compute the time complexity of the given algorithm. You **must** explain how you computed the time complexity and you **must** give the order (big-Oh) of the time complexity.

To explain how you computed the time complexity your answer **must** be like this: "Number of operations performed outside the `for` loop: x; number of operations performed in one iteration of the `for` loop: y; the number of iterations of the `for` loop is z **because** ... (the explanation **is very important**); total number of operations performed by the for loop: w. Total number of operations performed by the algorithm: x + w. The order of the time complexity is O(v)."

20. (6 marks)

```
private boolean compare() {
   int index1 = queue1.front;
   int index2 = queue2.front;
   boolean same = true;
   for (int i = 0; i < n; ++i) {
       if (queue1.array[index1] != queue2.array[index2]) same = false;
       index1 = (index1 + 1) % M;
       index2 = (index2 + 1) % M;
   }
   if ((index1 != queue1.rear) || (index2 != queue2.rear)) same = false;
   return same;
}
```

Outside the `for` loop (before the loop and after the loop) a constant number c_1 of operations is performed.

Each iteration of the `for` loop performs a constant number c_2 of operations. Notice that the `for` loop performs n iterations because the value of i is initially 0, at the end of each iteration the value of i increases by 1 and the loop ends when $i = n$. Hence, the total number of operations performed by the `for` loop is nc_2.

The total number of operations performed by the algorithm is then $c_1 + c_2 n$. Since $c_2 n$ asymptotically dominates c_1, the time complexity of this algorithm is $O(n)$.

21. (7 marks)

```
int x = n-1;
int z = 0;
for (int i = 0; i < n; i = i+x)
   for (int j = 0; j < n; j = j + 1)
       z = z + 1;
```

Outside the loop a constant number c_1 of operations is performed.

We analyze first the inner `for` loop. One iteration of this loop performs a constant number c_2 of operations. Every time that the first `for` loop is performed, the number of iterations of the second `for` loop is n because at the beginning $j = 0$, the value of j increases by 1 at the end of each iteration, and the loop ends when $j = n$. Hence, one execution of the second `for` loop performs $c_2 n$ operations.

Each iteration of the first `for` loop performs a constant number of operations c_3 to increase the value of i and compare i with n plus $c_2 n$ operations in the second `for` loop. Hence, one iteration of the first `for` loop performs $c_3 + c_2 n$ operations.

In the first iteration of the first `for` loop the value of i is set to zero. At the end of the first iteration the value of i increases by $x = n - 1$, so in the second iteration $i = n - 1$. At the end of the second iteration the value of i is increased again by x so i takes value $2n - 1$ and thus the `for` loop ends.

As the first loop iterates only 2 times, the total number of operations performed by the algorithm is $c_1 + 2(c_3 + c_2 n)$. The term $2c_2 n$ is the asymptotically dominating term, so the time complexity of this algorithm is $O(n)$.

22. (7 marks)

```
int i = 0, x = 1, y = 0;
while (i < n) {
   y = y + 1;
   if (y > x) i = i + n;
   else i = i + 1;
}
```

The number of operations performed outside the `while` loop is constant, c_1. These operations set $i = 0$, $x = 1$ and $y = 0$.

Each iteration of the `while` loop performs a constant number c_2 of operations. In the first iteration the value of y increases by 1, so $y = 1$. Since it is not true that $y > x$ the `else` statement is executed and i increases by 1, i.e. $i = 1$.

In the second iteration of the `while` loop the value of y increases again, so $y = 2$. Note that now $y > x$ so the value of i increases by n making $i = n + 1$. As now the condition of the `while` loop is false, the loop terminates. The total number of iterations of the `while` loop then is 2.

The total number of operations performed by the algorithm is $c_1 + 2c_2$, so the time complexity is $O(1)$.

For the following two questions write algorithms in Java or in detailed Java-like pseudocode like the one used in the lecture notes.

23. (16 marks) A palindrome is a string that reads the same forward and backward. For instance, the following string "`murder for a jar of red rum`" is a palindrome if we ignore the spacing. Complete method `isPalindrome` below so that it returns `true` if the string used as parameter (**ignoring the spacing**) is a palindrome and it returns false otherwise. You must use below queue `charQueue` and stack `charStack` in your solution. You **cannot** use any other auxiliary data structures. Method `text.substring(i,i+1)` returns a `String` object with the `i`-th character of `text`.

    The only queue operations that you can use are `enqueue`, `dequeue`, and `isEmpty`; the only stack operations that you can use are `push`, `pop`, and `isEmpty`. You **cannot** use any other data structures.

    You may assume that the first character and the last character of `text` are not spaces.

```
private boolean isPalindrome(String text) {
   LinkedQueue<String> charQueue = new LinkedQueue<String>();
   LinkedStack<String> charStack = new LinkedStack<String>();
   for (int i = 0; i < text.length(); i++) {
       charStack.push(text.substring(i,i+1));
       charQueue.enqueue(text.substring(i,i+1));
   }
   String charFromQueue, charFromStack;
   while (!charQueue.isEmpty()) {
       charFromQueue = charQueue.dequeue();
       while (!charQueue.isEmpty && charFromQueue.equals(" "))
           charFromQueue = charQueue.dequeue();
       charFromStack = charStack.pop();
       while (!charStack.isEmpty() && charFromStack.equals(" "))
           charFromStack = charStack.pop();
       if (!charFromStack.equals(charFromQueue)) return false;
   }
   return true;
}
```

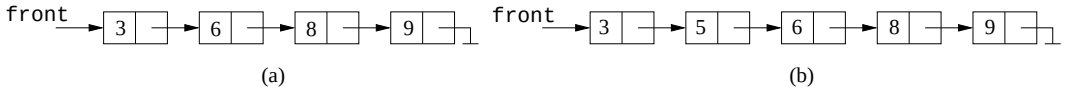
    A solution in pseudocode is given below.

```
private boolean isPalindrome(String text) {
   LinkedQueue<String> charQueue = new LinkedQueue<String>();
   LinkedStack<String> charStack = new LinkedStack<String>();
   for (int i = 0; i < text.length(); i++) {
       charStack.push(text.substring(i,i+1));
       charQueue.enqueue(text.substring(i,i+1));
   }

   while (!charQueue.isEmpty()) {
       charFromQueue = charQueue.dequeue()
       while (!charQueue.isEmpty and (charFromQueue is " "))
           charFromQueue = charQueue.dequeue()
       charFromStack = charStack.pop()
       while (!charStack.isEmpty() and (charFromStack is " "))
           charFromStack = charStack.pop()
       if (charFromStack ≠ charFromQueue) return false
   }
   return true
}
```

24. (16 marks) Consider a singly linked list formed by node objects of class `LinearNode`. This class provides methods `getNext()` and `setNext(next)` to get and set the next nodes in the list, and `getValue()` to get the value stored in a node; `LinearNode(x)` creates a new node storing value `x`. Each node stores an integer value and the values are stored in the nodes in increasing order, so the first node stores the smallest value, the second node stores the second smallest value, and so on. Let `front` be a reference to the first node in the list.

Write an algorithm `insert(x)` that adds a new node storing the integer value `x` into the list in the proper position so the values in the list appear in increasing order. You might assume that the list is not empty, that there are no duplicated values in the list, and that `x` is different from all the values in the list. For example if the list is as list (a) below and we execute `insert(5)` then the resulting list must be as in list (b).



(a) (b)

Algorithm `insert (x)`

```
newNode = new LinearNode(x)
current = front
previous = null
while (current ≠ null) and (current.getValue() < x) do {
    previous = current
    current = current.getNext()
}
if current = front then {
    newNode.setNext(front)
    front = newNode
}
else {
    previous.setNext(newNode)
    newNode.setNext(current)
}
```

The algorithm in Java is given below.

```
public void insert (x) {
   LinearNode newNode = new LinearNode(x);
   LinearNode current = front;
   LinearNode previous = null;
   while ((current != null) && (current.getValue() < x)) {
       previous = current;
       current = current.getNext();
   }
   if (current == front) {
       newNode.setNext(front);
       front = newNode;
   }
   else {
       previous.setNext(newNode);
       newNode.setNext(current);
   }
}
```

Western University
Department of Computer Science

CS1027b Foundations of Computer Science II
Midterm Exam

March 10, 2018

PART I	
PART II	
18	
19	
20	
21	
22	
23	
24	
Total	

Last Name: _____

First Name: _____

Student Number: _____

Section Number (1 - Solis-Oba, 2 - Beauchemin): _____

Instructions

- Fill in your name, student number, and section.
- The exam is 2 hours long and it has a total of 100 marks.
- The exam has 11 pages and 24 questions.
- The first part of the exam consist of multiple choice questions. For each question circle only **one** answer.
- For the second part of the exam, answer each question only in the space provided.
- When you are done, raise your hand and one of the TA's will collect your exam.