Print your name:______________________________________________
Student number: _______________________________________

Instructions:

- Fill in your name and student number above immediately.
- Answer all the questions in the exam in the spaces provided.
- You have 2 hours to complete the exam.
- The exam has 8 questions on 15 pages, and is out of a possible 80 marks.
- The marks for each individual question are given. Allow approximately 1.5 minute per mark on average.
- There is a page for rough work at the end of the exam paper.
- Read the questions carefully!

DO NOT TURN THIS PAGE UNTIL DIRECTED TO DO SO.

<table>
<thead>
<tr>
<th>Question</th>
<th>Max Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>TOTAL</td>
<td>(max 80)</td>
</tr>
</tbody>
</table>
1. (15 marks) True/False (Circle your answers)

   a) (1 mark) If two object references are equal in state, then they refer to the same object.
      True  False

   b) (1 mark) The time complexity of an algorithm consists of the most dominant term of its growth function.
      True  False

   c) (1 mark) Polymorphism allows a reference variable to point to objects of unrelated types.
      True  False

   d) (1 mark) Generic types are bound at compile time.
      True  False

   e) (1 mark) An interface can be used even when it is not implemented by another class.
      True  False

   f) (1 mark) Data encapsulation requires that instance variables be declared as private.
      True  False

   g) (1 mark) A time complexity of $O(n)$ for a program means that it will run in constant time.
      True  False

   h) (1 mark) Inherited variables and methods can be used in a derived class as if they had been declared locally.
      True  False
i) (1 mark) The child of a class can be the parent of one or more classes.
   True  False

j) (1 mark) All Java classes are derived, directly or indirectly, from the Object class.
   True  False

k) (1 mark) Polymorphism is made possible through class inheritance.
   True  False

l) (1 mark) A static method within a class cannot be invoked without having to instantiate an object of that class.
   True  False

m) (1 mark) A Java interface should never have a constructor.
   True  False

n) (1 mark) A Java interface never contains an attribute (instance variable) declaration.
   True  False

o) (1 mark) A new class X derived from a class Y establishes an is-a relationship from class X to class Y.
   True  False
2. (5 marks) The following inheritance diagram represents a hierarchy of Java classes. Use the diagram to determine whether the statements given below it are True or False.

![Inheritance Diagram](image)

a) (1 mark) Class_G implements Class_C.

True  False

b) (1 mark) Class_F extends Class_A.

True  False

c) (1 mark) Suppose that Class_C is the only class in the diagram that contains a public method `blah()` . If a Java program attempts to invoke `blah()` against an object from Class_F, the method in Class_C will be executed.

True  False

d) (1 mark) Again, suppose that Class_D is the only class in the diagram that contains a public method `blah()` . If a Java program attempts to invoke `blah()` against an object from Class_A, the method in Class_D will be executed.

True  False

e) (1 mark) Suppose that we have the declaration `Class_A a; ` The variable a can hold a reference to an object from Class_I.

True  False
3. (10 marks) Give the output generated by each of the following Java code segments:

a) (2 marks) final int MAX = 10;
   for (int i = 0; i < MAX; i+=2)
   System.out.print(i + " , ");

   Answer: 0, 2, 4, 6, 8,

b) (2 marks) LinkedQueue<Integer> a = new LinkedQueue<Integer>();
   final int MAX = 5;
   for (int i = 0; i < MAX; i++) a.enqueue(MAX - i);
   System.out.println(a.dequeue());

   Answer: 5

c) (2 marks) LinkedQueue<Integer> a = new LinkedQueue<Integer>();
   LinkedStack<Integer> b = new LinkedStack<Integer>();
   final int MAX = 5;
   for (int i = 0; i < MAX; i++) {
       a.enqueue(i);
       b.push(i);
   }
   for (int i = 0; i < MAX; i++)
       System.out.println(a.dequeue() + b.pop() + " , ");

   Answer: 4, 4, 4, 4, 4,

d) (2 marks)

   final int MAX = 32;
   for (int i = 1; i <= MAX; i*=2) System.out.print(i + " , ");

   Answer: 1, 2, 4, 8, 16, 32,
e) (2 marks)

```java
LinearNode<Integer> a = new LinearNode<Integer>(1);
LinearNode<Integer> b = new LinearNode<Integer>(2);

a.setNext(b);
b.setNext(a);

System.out.println(a.getNext().getElement());

Answer: 2
```
4. (15 marks) In each of the following situations, use **Big-O notation** to express the amount of work being done in terms of \( n \).

a) (1 mark) Java code is used to print out the value of the last element in an array of \( n \) elements.

   **Answer:** \( O(1) \)

b) (2 marks) Java code is used to print out the value of the last element in a queue containing \( n \) elements (assume it is a linked queue).

   **Answer:** \( O(1) \)

c) (2 marks) A Java program takes \( n^2 \log n + n^2 \) steps to complete, given an input of size \( n \). What is its time-complexity, in big-O notation?

   **Answer:** \( O(n^2 \log n) \)

d) (2 marks) The following Java method is executed:

   ```java
   public static void print1( int [ ] list, int n ) {
       for ( int i = 0; i < n; i++ )
           System.out.println( list[i] );
   }
   ```

   **Answer:** \( O(n) \)

e) (2 marks) The following Java code segment is executed:

   ```java
   int k = 0;
   for (int i = 1; i <= n; i++) {
       for (int j = 1; j <= n; j++) {
           k = i*j;
           System.out.println(k);
       }
   }
   ```

   **Answer:** \( O(n^2) \)
f) (2 marks) The following Java code segment is executed:

```java
int i = n;
while (i > 0) {
    System.out.println(i);
    i = i/2;
}
```

Answer: $O(\log_2 n)$

---

g) (2 marks) The following Java method is executed:

```java
public static void havingFun(int n) {
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            for (int k = 0; k < n; k++)
                System.out.println(i,j,k);
}
```

Answer: $O(n^3)$

---

h) (2 marks) The following Java method is executed:

```java
public static int someSum(int[][] table, int n) {
    int sum = 0;
    for (int k = 0; k < n; k++)
        sum = sum + table[n-k-1][n-k-1];
    return sum;
}
```

Answer: $O(n)$
A palindrome is a sentence that reads the same forward and backward. For instance the following sentence: *murder for a jar of red rum* is a palindrome, if we ignore the spacing. Complete the method `isPalindrome` below in such a way as to determine if the characters given by the user (assume there is no spacing) is a palindrome. The method receives a stack and a queue, containing the letters from the user. Use both of these data structures to determine if the text is a palindrome.

```java
import java.util.Scanner;

public class Palindrome {
    private static Boolean isPalindrome(LinkedStack<String> CharStack, LinkedQueue<String> CharQueue) {
        String c1, c2;
        while (!CharStack.isEmpty() && !CharQueue.isEmpty()) {
            c1 = CharStack.pop();
            c2 = CharQueue.dequeue();
            if (!c1.equals(c2)) return false;
        }
        return true;
    }

    public static void main(String[] args) {
        String text;
        Scanner in = new Scanner(System.in);
        System.out.print("Enter text: ");
        text = in.nextLine();
        int length = text.length();
        LinkedQueue<String> CharQueue = new LinkedQueue<String>();
        LinkedStack<String> CharStack = new LinkedStack<String>();
        for (i = 0; i < length; i++) {
            CharStack.push(text.substring(i, i + 1));
            CharQueue.enqueue(text.substring(i, i + 1));
        }
        System.out.println(isPalindrome(CharStack, CharQueue));
    }
}
```
[This page was intentionally left blank]
6. (10 marks) Rewrite the `size` method from the `LinkedStack.java` class, assuming that the instance variable `count` does not exist.

```java
public int size(){
    int num = 0;
    LinearNode<T> curr;
    curr = this.top;
    while (curr != null) {
        num++;
        curr = curr.getNext();
    }
    return num;
}
```
7. (10 marks) Assume the instance variable `count` in the class `CircularArrayQueue` does not exist. Program the `size` method, using the values of `front` and `rear`, to compute the number of elements in the circular queue.

```java
public int size() {
    if (rear == front) {
        if (queue[rear] == null) return 0;
        else return queue.length;
    }
    else {
        if (front < rear) return rear - front;
        else return queue.length + rear - front;
    }
}
```
8. (5 marks) What is the output produced by the following program?

```java
public class BuildLinkedList {
    public static void main(String[] args) {
        final int MAX = 5;

        LinearNode<Integer> head = null;
        LinearNode<Integer> intNode;

        for (int i = MAX; i >= 1; i--)
        {
            intNode = new LinearNode<Integer>(MAX - i + 1);
            intNode.setNext(head);
            head = intNode;
        }

        LinearNode<Integer> current = head;
        for (int i = 1; i <= MAX; i++)
        {
            System.out.print(current.getElement() +", ");
            current = current.getNext();
        }
    }
}
```

Answer: 5, 4, 3, 2, 1,