Please write your entry number and name on each sheet.

Entry Number: **Name:**

COL 106 MINOR EXAM I

SEMESTER I 2019-2020

1 hour

A

Please do not allow any bag, phone or other electronic device near you. Keep your ID card next to you on the desk. Max marks for questions are listed in* []. *Write answers in the provided space.

1. [16]

a) [6] In the following code fragment, when C.main() is called, how many objects of each class are created? (For this purpose, do not include the objects of class B in the count for class A.) What is the reference count for each object? Explain.

```
class A {}
class B extends A {
    A a;
    B b;
    B() {
        A ta = new A(); b = ta; B b = (B) a;
    }
}
class C {
    static void main() {
        B b1; B b2 = new B();
    }
}
```

No. of objects of class A:
Reference counts of each:

No. of objects of class B:
Reference counts of each:

No. of objects of class C:
Reference counts of each:

b) [2] In the following example, method **m** is overridden as well as overloaded in class **B**, while variable **v** is hidden. What does that mean?

```
class A {
   int v;
   void m() { v = 10;}
}
class B extends A {
   int v;
   void m() { v = 5;}
   void m(int p} { v = p;}
}
class C {
   void main() {B b = new B(); b.m();}
}
```

c) [2] Explain the effect if the first method **m()** above in the class B is replaced with

void m() { super.v = 5; }

What is the value of b.v after the call to "b.m()" at the end of main().

d) [2] Just like there is a **`super`** keyword used in the previous example, why is there no corresponding **`child`** keyword?

e) [2] Why ever create a subclass at all?

f) [2] Why must the parent's constructor be called in every class's constructor?

2. [8] Prove that $f(n) = n\log^2 n + n^2 \log n + n^3$ is $\Theta(n^3)$

3. [4] What is the *postcondition* for the following statement if the *precondition* is "True"

```
if(x<0) abs = -x; else abs = x;
```

4. [8] Prove by the contra-positive method:

a) [4] If n is a positive integer such that n%4 is either 2 or 3, then n is <u>not</u> a perfect square.

List the contra-positive statement:

then prove it:

Please write your entry number and name on each sheet.

Entry Number: **Name:**

b) [4] If x and y are two integers s.t. $x*y$ is even, then at least one of the two must be even.
List the contra-positive statement:
then prove it:

5. [8] **Prove** that the following algorithm finds the smallest integer in an array of n ints. **Analyse** its worst case complexity in big-**Θ** terms.

```
min = A[0]
for i = 1 until n-1
    if(A[i] < min) min = A[i]
```

6. [8] The following binary search pseudo-code finds **v** in array **A** between **lo** and **hi** (inclusive). **Prove** that it indeed finds the index for **v**, if it exists, and returns FAIL, if it does not. **Analyse** its worst case complexity in big-**Θ** terms – write the recurrence relation and solve it.

```
BinSearch(A, lo, hi, v):
        if(lo > hi) return FAIL;
mid = ⌊(lo+hi)/2⌋
if(A[mid] == v) return mid;
if(A[mid] < v) return BinSearch(A, mid+1, hi, v);
else            return BinSearch(A, lo, mid-1, v);
```

7. [2] Threads of a process share the heap, but maintain their own stacks. **Explain**. (List which variables are shared and which are not.)

8. [8] The following code tries to ensure, without the help of the synchronized keyword, that two threads never execute the function exclude concurrently. It does so by wrapping it in the function properexclude. Does this code succeed? If so, **prove** it. If not, **describe** the problem(s) and **provide fix** it with minimal changes. Do not use synchronized.

```
class Twoway {
    static int turn = 0;
    private int id;
    Twoway(int id) { this.id = id; }
    void properexclude() {
        if(turn == id) {
            exclude(); // Assume it exists
        }
        turn = (id==1)? 0:1; // Other's id
    }
}
```

Two threads are initialized as follows:

```
    Thread t1 = new Thread(new Twoway(0));
    Thread t2 = new Thread(new Twoway(1));
```

Both thtreads may call exclude multiple times.

Please write your entry number and name on each sheet.

Entry Number: **Name:**

9. [8] The following code manages a (singly) linked list. What if multiple threads can share the reference to the same list. **Insert** synchronized on appropriate lines so that a shared list is never corrupted. Only synchronize on nodes.

```
1 class Node<T>  {
2    Node<T> next;
3     T value;
4    Node(Node<T> n, T v) {
5          next = n; value = v;
6    }
7 }

9   class LinkedNode<T> {
10     private Node<T> sentinel =
11               new Node<T>(null, null);
12    Node<T> addafter(Node<T> node, T val) {
13       Node<T> newnode =
14               new Node<T>(node.next, val);
15       node.next = newnode;
16       return newnode;
17    }
18    Node<T> addfirst(T val) {
19       return addafter(sentinel, val);
20    }
21     Node<T> deleteafter(Node<T> node) {
22       // Never delete after the last node
23       node.next = node.next.next
24    }
25 }
```

Overflow sheet. Do not remove this sheet. No other sheet will be provided.