COL106 Minor 1 - Set B

ARPIT SAXENA

TOTAL POINTS

53 / 70

QUESTION 1

1 16 pts

1.1 1 (a) 5 / 6

- **1 pts** object count of A incorrect : object ta is created of class A

✓ **- 1.5 pts reference count of A incorrect : 0 reference count for this object ta as the object is defined in the constructor of B and hence can't be accessed after constructor has been called.**

- **0.5 pts** no explanation for part A reference count

- **1 pts** object count of B incorrect: Only b2 of class B is created

- **1.5 pts** reference count of B incorrect: only reference count that is present is b2.

- **0.5 pts** no explanation for part B reference count

- **0.5 pts** object count of C incorrect: clearly no object of class C is created

- **0.5 pts** reference count of C incorrect: no object created and hence no reference count

- **6 pts** all incorrect/blank

+ 0.5 Point adjustment

1.2 1 (b) 2 / 2

✓ **- 0 pts All correct**

- **0.67 pts** overridden definition is incorrect : m() is redeclared with the same signature as in the parent class A. This overrides the base class's method m()

- **0.66 pts** overloaded definition is incorrect. In derived class B, an additional method m() is declared, which has a signature different from the other available m().

- **0.67 pts** Variable v is hidden is incorrect; A child class can define the variable with the same name present in the superclass. This hides the otherwise directly usable variable v declared in the parent's class, which is now accessible only through the super keyword.

- **2 pts** Blank or incorrect

+ **0.5 pts** incomplete explanation

- **0.5 pts** not mentioning overloading and overriding saparately

1.3 1 (c) 2 / 2

✓ **+ 1 pts Set v of class A instead v of class B**

✓ **+ 1 pts Correct value of v (unmodified/0)**

+ **0 pts** Unattempted/Incorrect

1.4 1 (d) 1 / 2

+ **2 pts** Parent exists independently , Not supposed to know about children

✓ **+ 1 pts Ambiguity : Parent can have multiple children with same attributes**

+ **0 pts** Unattempted/Incorrect

1.5 1 (e) 2 / 2

✓ **+ 1 pts Reuse of code**

+ **1 pts** Add difference on top of parent class

+ **1 pts** Abstraction of details

✓ **+ 1 pts Modular design**

+ **0 pts** Unattempted/incorrect

1.6 1 (f) 0 / 2

+ **1 pts** Separation of concern - parent handles its members, child handles its own. Subclass contains fields and methods from superclass and by calling constructor of parent class we can make sure that these fields are initialized.

+ **1 pts** Only parent's constructor can initialize private variables belonging to it

✓ **+ 0 pts Unattempted/Incorrect**

QUESTION 2

2 2 4 / 8

✓ **+ 2 pts** **Identify volatile keyword missing for shared variable**

+ 2 pts Identify progress condition violations

+ 4 pts Proves mutual exclusion under the assumption of volatile memory access. The grade may be decreased depending on the quality of proof or missing arguments.

+ 4 pts Provided fix ensuring mutual exclusion, progress, and deadlock freedom

+ 3 pts Fix solves mutual exclusion but either progress or deadlock freedom is missing

✓ **+ 2 pts** **Incomplete fix - but towards the right direction and solves at least one among mutual exclusion, progress, or deadlock freedom.**

+ 2 pts Used locks to provide correct solution

+ 1 pts Used synchronized to provide correct solution.

+ 0 pts No issues identified correctly, or no fixes provided. Or, the fix doesn't solve any of the problems.

Note that scheduling only one of the threads is not a correct solution. Also, if you claim the success of this code, then the proof must be there.

QUESTION 3

3 3 4 / 8

+ 0 pts Not attempted/Incorrect

+ 3 pts Sync on addafter and deleteafter with explanation

✓ **+ 4 pts** **Single synchronize in addafter and deleteafter**

+ 8 pts All Correct

+ 6 pts Synchronize on incorrect nodes

+ 2 pts Partial marks

- 1 pts No/Incorrect explanation

QUESTION 4

4 4 1.5 / 2

✓ **+ 1.5 pts** **Correct explanation**

✓ **+ 0.5 pts** **Listed variable**

+ 0 pts Wrong

+ 0 pts Not attempted

- 0.5 **Point adjustment**

QUESTION 5

5 8 pts

5.1 5 (a) 4 / 4

✓ **+ 2 pts** **Correct contrapositive statement**

- 0.5 pts Not mentioned that n is positive integer in the contrapositive statement / incorrectly mentioned

+ 2 pts Proving contrapositive is true for all exhaustive cases. Equal weightage for each case.

- 0.25 pts While writing n=k square, k must be integer.

- 0.5 pts Did not express n in '4(z) + remainder' form and simply concluded about value of n%4.

- 0.25 pts While expressing k as 4p/4p+1/4p+2/4p+3 etc, did not mention what is domain of k/ incorrectly mentioned.

+ 0 pts Incorrect

- 0.25 pts contrapositive stmt partially correct

+ 2 **Point adjustment**

why have you squared . giving marks

5.2 5 (b) 4 / 4

✓ **+ 2 pts** **Listing the contra-positive statement correctly**

- 0.5 pts Not mentioned that x and y are integers in the contrapositive statement/ Incorrectly mentioned

✓ **+ 0.5 pts** **Represent x and y in 2k+1/2m+1 form where x and y are integers and give reason why it can be done.**

+ 2 pts Correct proof

✓ **+ 0.5 pts** **Writing x*y in product form : (2k+1)(2m+1)**

✓ **+ 1 pts** **Expansion of product and then showing x*y can be written as 2p+1 for some p.**

- 0.25 pts while expressing x/y as 2k+1, domain of k is missing/ incorrect.

- 0.25 pts Not mentioning why x/y can be written as 2k+1/2m+1

+ 0 pts Incorrect

+ **1 pts** patrially correct proof.. no 2k+1 form and product expansion

QUESTION 6

6 6 **8 / 8**

✓ + **2 pts** **Loop Invariant**

- **1 pts** Loop Invariant stated indrectly (implied)

✓ + **1 pts** **Base case/Initialisation/Pre-condition**

✓ + **1 pts** **Induction hypothesis/Maintenance(State)**

✓ + **1 pts** **Induction Step/Maintenance(Reason)**

✓ + **2 pts** **Complexity Argument**

- **1 pts** Not stating the number of comparisons

✓ + **1 pts** **Complexity Analysis**

+ **0 pts** Incorrect or Unattempted

QUESTION 7

7 7 **7.5 / 8**

✓ + **1 pts** **Induction Base Case**

✓ + **1 pts** **Correct Induction hypothesis**

✓ + **2 pts** **Correct Inductive Step**

✓ + **2 pts** **Correct Recurrence Relation**

✓ + **2 pts** **Correct Recurrence Solution**

+ **0 pts** Incorrect or Not Attempted

- **0.5** **Point adjustment**

You have directly written log n times. Why log n times? n/2^i =1 , i.e. i = log n....

QUESTION 8

8 8 **4 / 8**

+ **4 pts** Proving Big O

+ **4 pts** Proving Big Omega

+ **7 pts** Proving using limits

- **1 pts** Inequality used is not stated or not correct for Big O

- **1 pts** Final inequality is not complete (constant not mentioned) in Big Omega

- **1 pts** Inequality used is not stated for Big Omega

- **1 pts** Final inequality is not complete (constant not mentioned properly or inequality incorrect) in Big O

+ **0 pts** Unattempted

+ **0 pts** Incorrect

+ **4** **Point adjustment**

QUESTION 9

9 9 **4 / 4**

✓ + **4 pts** **Correct Post Condition**

+ **0 pts** Unattempted/Incorrect

Please write your entry number and name on each sheet.
Entry Number: 2018MT10742 **Name:** ARPIT SAXENA

COL 106 MINOR EXAM I

SEMESTER I 2019-2020

1 hour

B

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: 1

Reference counts of each: 2 (ta, b)

constructed in B's constructor, assigned to ta.
the ~~assigned~~ referenced by b (that we would throw a compiler error as subclass refs. can't refer to their parent's objects, but still counts)

No. of objects of class B: 1

Reference counts of each: 1 (b2)

only b2 refers to the new object that is created in C.main(). b1 is null.

No. of objects of class C: 0

Reference counts of each: -

No object of C is created.

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();}
}
```

- m being overriden means that the method m() in B has replaced the method m() defined in A. It has been dynamically bound to an object of B, and will be called even when called through a reference of A
- overloading means there are two functions of same name but with different signature, so when ~~m()~~ m is called the signature is matched and appropriate fn. is called
- v has been hidden in class B. class A also has a v, but since B also declares a v, all statements in B ~~a~~ mentioning v will refer to v declared in B.

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().

super.v = 5 assigns 5 to A's ~~constructe~~ ~~to~~ v.
b.v would be 0 ~~after~~ at end of main. (Primitives have a value 0 if not explicitly declared)

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

This is because a parent class can have multiple child classes, in which case child would be ambiguous.
~~A~~ parent of a class, if it exists, has to be unique.

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

A subclass helps in code reuse as one doesn't have to write the code again if a specific function is required. It also allows us to model real-world scenarios where one object is also another type of object.

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

This is because an object of type B is also an object of type A, and so inherits A's methods (which are not private). These methods can also be called by methods in B. The methods of A, however, expect A to be properly constructed via the constructor. So, the parent constructor must always be called so that the class invariant can be established on which all its methods depend

2. [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
    }
}
```

The problem is that this ~~id~~ turn variable might be cached so not visible to other thread.

solution is to declare it volatile ~~so~~

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.

3. [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.

```
class Node<T> {
  Node<T> next;
   T value;
   Node(Node<T> n, T v) {
        next = n; value = v;
   }
}

class LinkedNode<T> {
    private Node<T> sentinel =
                new Node<T>(null, null);
    Node<T> addafter(Node<T> node, T val) { synchronize (node) {
      Node<T> newnode =
                new Node<T>(node.next, val);
      node.next = newnode; }
      return newnode;
    }
    Node<T> addfirst(T val) {
      return addafter(sentinel, val);
    }
     Node<T> deleteafter(Node<T> node) { synchronize (node) {
      // Never delete after the last node
      node.next = node.next.next ;
    }}
}
```

Please write your entry number and name on each sheet.
Entry Number: 2018MT10742 **Name:** ARPIT SAXENA

4. [2] Threads of a process share the heap, but maintain their own stacks. **Explain.** (List which variables are shared and which are not.)

Threads can have references, which point to the same object as they share the heap on which the objects are allocated. The reference variables, which lie on stack, are not shared. Also, primitives (int, float, etc.) are also not shared.

5. [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 not a perfect square.
List the contra-positive statement: If n is a positive integer such that n is ~~either~~ a perfect square, then n%4 is either 0 or 1
then prove it:

Let n be a positive integer such that n is a perfect square.
n can either be a square of an even or an odd ~~integer~~ number.

i) $n = 2k, \Rightarrow n^2 = 4k^2 \Rightarrow n^2 \% 4 = 0$
$k \in \mathbb{N}$

ii) $n = 2k+1, k \in \mathbb{N} \Rightarrow n^2 = (2k+1)^2 = 4(k^2+k)+1 \Rightarrow n^2 \% 4 = 1$

$\Rightarrow n^2 \% 4$ can be either 0 or 1
or $n^2 \% 4$ can neither be 2 nor 3

Hence, proved.

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: If x and y are two integers s.t. both are odd, then $x*y$ is odd
then prove it:

Let x and y be two integers which are both odd

$\Rightarrow x = 2m+1, y = 2n+1$; m, n are integers

$\Rightarrow x*y = (2m+1)(2n+1)$
$= 4mn + 2m + 2n + 1$
$= 2(2mn + m + n) + 1$
$= 2k+1$, where $k = 2mn + m + n \in \mathbb{Z}$

$\Rightarrow x*y$ is odd.

Hence, proved.

6. [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]
```

The for loop has the following invariant ~~that~~:

$\min = \min(A[0..i-1])$, where $\min(A)$ function is the min. of elements of A.

$1 \le i \le n$

Initialisation: $i = 1$ and $\min = A[0] = \min(A[0..1-1])$

⇒ Loop invariant is true before the first iteration.

Maintenance: Suppose that the loop invariant is true ~~after~~ before the ith iteration i.e.

$\min = \min(A[0..i-1])$

Then $A[i] < \min \Rightarrow \min = A[i]$. If $A[i] < \min(A[0..i-1])$ then $A[i]$ is minimum of $A[0..i-1]$.

Else, it is not.

$\therefore \min = \min(A[0..i])$

⇒ Invariant is true after ~~(i+1)th~~ ith iteration.

Termination: Loop terminates when $i = n$. Invariant ⇒ $\min = \min(A[0..n-1]) = \min(A)$

⇒ The algorithm is correct (Note that loop must terminate as i is increased by 1 in each iteration)

The loop runs n-1 times. In worst case, two statements are executed with in it each time

⇒ Running time, $T(n) = 1 + (n-1) * 2 = 2n-1 = \Theta(n)$ ⇒ Worst case Running time $= \Theta(n)$

7. [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);
```

We want to prove that $P(n)$: BinSearch finds index for v between lo and hi (inclusive), if it exists, and returns FAIL, otherwise, $n = hi - lo + 1$

~~Let that~~ We have that using induction on $n = hi - lo + 1$

Base case: $n = 0$ ⇒ $lo = hi + 1 \Rightarrow lo > hi$, ⇒ v does not exist

So, FAIL is returned

Induction Hypothesis: Assume $P(k)$ is true $\forall k \le n$

Induction step: If $hi - lo + 1 = n + 1$, the $mid = (hi + lo)/2 \le n$.

If $A[mid] == v$, then we have found v and return it

Else, since A is sorted, if $A[mid] < v$, then v, if it exists, must lie between mid+1 to hi, ~~which is less than~~ → which finds correctly (by induction hypothesis).

Similar for when $A[mid] \ge v$.

In the worst case, v is not present in A. Let T(n) be running time

Then, $T(n) = T(n/2) + \Theta(1)$. {T(n/2) is from further BinSearch calls}

⇒ $T(n) = T(n/2) + b$, for some $a, b \in \mathbb{R}$

$T(n/2) = T(n/4) + b$ ⇒ $T(n) = T(n/4) + 2b$

Continuing, $T(n) = T(n/2^k) + a\left(n + \frac{n}{2} + \cdots + \frac{n}{2^{k-1}}\right) + b$. When $k = \lceil \log n \rceil$, $T(n/k) = 0$

⇒ $T(n) = b\lceil \log n \rceil$

⇒ $T(n) = \Theta(\log n)$

Please write your entry number and name on each sheet.

Entry Number: 2018MT10742 **Name:** ARPIT SAXENA

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

(i) $f(n) = O(n^3)$. Let $k = 100$, then

$$n\log^2 n + n^2\log n + n^3 \leq 100n^3 \quad \forall n \geq 1$$

$\Rightarrow f(n) = O(n^3)$

We observe that it is true for $n=1$. Let $f(x) = 100x^3 - (x^3 + x\log^2 x + x^2\log x)$, ~~100~~

$f'(x) = \cancel{300x}^2 3 + 99x^2 + \cancel{x}(2\log x) + \log^2 x + 2x\log x + x > 0 \quad \forall x > 1$

(ii) Let $k = 0.001$, then

$$n\log^2 n + n^2\log n + n^3 \geq 0.001n^3 \quad \forall n \geq 1$$

Observe it is true for $n = 1$

Let $f(x) = x\log^2 x + x^2\log x + x^3 - 0.001x^3$

$\Rightarrow f'(x) = \cancel{x}(2\log x) + \log^2 x + x^2 + 2x\log x + 3 \times 0.999x^2 > 0$

$\forall x > 1$

$\Rightarrow f(x)$ is increasing and $f(1) > 0$

$\Rightarrow n\log^2 n + n^2\log n + n^3 \geq 0.001n^3 \; \forall n \geq 1$

9. [4] What is the *postcondition* for the following statement if the *precondition* is "True"

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

postcondition : $abs = |x|$, where $|x| = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases}$

~~Or we can write $x < 0 \Rightarrow abs = -x$~~

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