PLC Final Answer Sheet

1. <https://github.com/SafwanA12/PLCFinalExam>

2. Programming Language Concepts 11th Edition

Boolean BNF :

**<Boolean\_expr> -> <Boolean\_expression> || <Boolean\_term> | <Boolean\_term>**

**<Boolean\_term> -> <Boolean\_term> && <Boolean\_factor> | <Boolean\_factor>**

**<Boolean\_factor> -> id | ! <Boolean\_factor> | ( <Boolean\_expr> )  | <relation\_expr>**

**<relation\_expr> -> id = = id | id != id | id < id | id <= id | id >= id | id > id|i**

3.

4. The four criteria are:

1: P=> I: the loop invariant must be true initially

2:{I and B} S{I}: loop invariant is not changed by executing body of loop

3:{I and (not B)} => Q: If I is true and B is false then Q is true

4: Loop terminates

5. <https://github.com/SafwanA12/PLCFinalExam>

(My readability on this code is almost the same except I have replaced the goto statements with a break statement which in this case will serve the same purpose in reaching the solution)

6. <https://github.com/SafwanA12/PLCFinalExam>

7. Tombstones definitely take up a lot of space and time due to the nature of it never deallocating storage. However I’m not sure if I can say that one method takes up more usage and time than the other, lets look at tombstones first, Heaps for example every time in a tombstone when a variable is accessed this adds another cycle to the machine increase the time/space. In terms of security the pointers in tombstones point to the tombstones instead of the memory holding the actual value which helps out to stop when someone is trying to steal deallocated memory information because of the fact the tombstone never changes. Locks-and-keys approach expresses the pointers as ordered pairs of keys and address. Heap-dynamic variables are assumed as the storage for the variable and a cell holding an integer. When a Heap Dynamic variable is allocated a lock value is also created in the heap dynamic variable, the lock cell, and the key cell of the pointer. Matching values are allowed access. In the event of mismatched values, it returns a run-time error. When it’s pointer dereferenced, this address value will still be not change, but the key value will not work anymore, so access is denied. In terms of time and space it is similar to Tombstone method.

8.

8a. <https://github.com/SafwanA12/PLCFinalExam>

8b. <https://github.com/SafwanA12/PLCFinalExam>

8c. <https://github.com/SafwanA12/PLCFinalExam>

8d. j is set to -3, it then will go on to the for loop. Then the program goes inot the switch statement followed by the different cases to enter by comparing the value of j + 2 to values. If j +2 == 2 or 3 the switch statement executes j. If j + 2 == 0 then j+=2 is executed. If after checking all other cases none of the cases match the value then it will default j to 0.Then the program breaks out of the loop, then if j is greater than or equal to 0 then j will be set to 3-i, then loop will continue to loop until i reaches the for loop condition of being equal to 3.

9. Typescript is a opensource programming language introduced in 2012 by Microsoft, which is in easier terms an addition to JavaScript that allows static typing, in my opinion if someone enjoys Java but hates the extra rules about specifying types and what not, they will love Typescript. Typescript **writability** is easier compared to Java. Typescript **readability** it slightly harder since every reference isn’t being specified. Typescripts **reliability** I would say is same level as Java due to it being created to serve the purpose of making large scale application except Java can be run on most machines while Typescript is a scripting language so it’s web browser/application based. When it comes to **logic and expressions with binary,unary, and trinary**, both work the same as they are used as modern day programming languages. Now if typescript were an older not as popular language such as Assembly it would be a different story on how expressions such as 2\*2 or 2+2 is to be written and then handled. In Java every variable and **expression** should have the same type. An example of this is the advantage typescript has over Java is a better **type and assignment** inference system where instead of having to specify an int is an int the language will assume “red = 12;” means were assigning the value of red to the integer 12 instead of having to specify in this way “int red = 12;” But there will be differences between Java and typescript due to Java being an object oriented programming language while typescript is a object based scripting language. Unlike Java in typescript you can use **keywords**  as an identifying name but because keywords are already registered into the Java programming language that is not possible.

Cited Webpages for 9:

<https://www.typescriptlang.org/docs/handbook/basic-types.html#introduction>

<https://www.beyondjava.net/comparing-typescript-java>