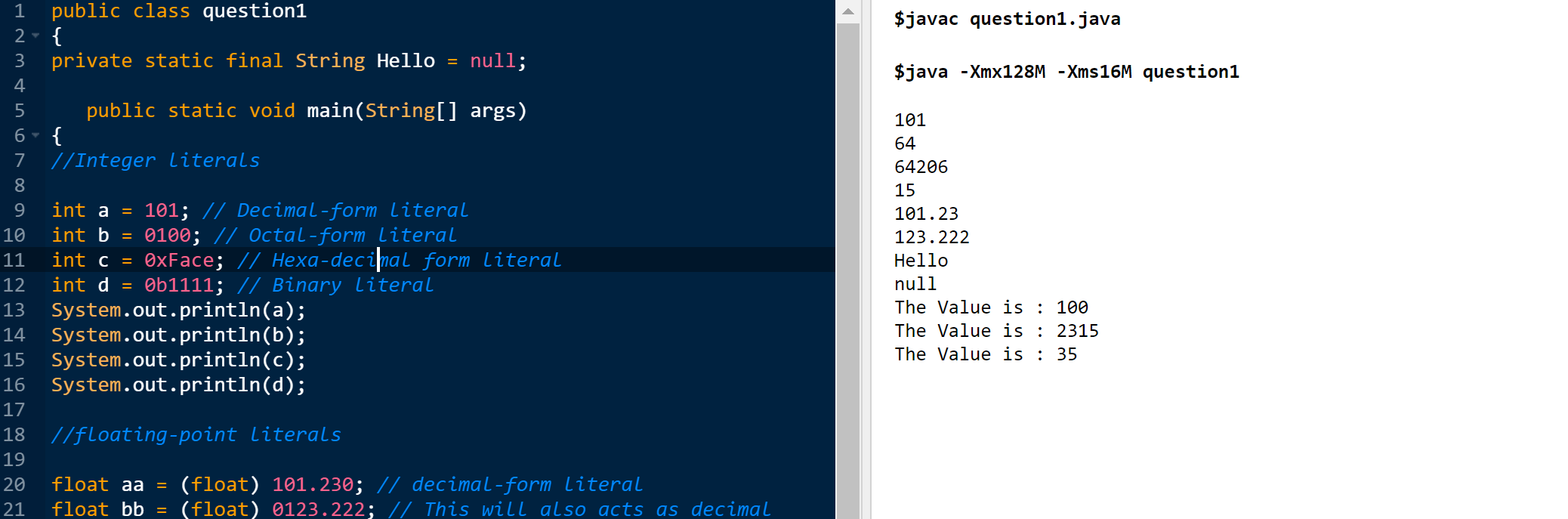
PLC Final Hunter Durand

1. **Question 1**



1. **Question 2: Also, in Github wrote a CNF for JAVA**

Target = MainClass, { ClassDeclaration }, EOF; /\* EOF:token returned by the scanner at end-of-file \*/ MainClass = "class", Identifier, "{", "public", "static", "void", "main", "(", "String", "[", "]", Identifier, ")", "{", Statement, "}", "}"; ClassDeclaration = "class", Identifier, [ "extends", Identifier ], "{", { VarDeclaration }, { MethodDeclaration } "}"; VarDeclaration = Type, Identifier, ";"; MethodDeclaration = "public", Type, Identifier, "(", [ Type, Identifier, { ",", Type, Identifier }, ], ")", "{", { VarDeclaration }, { Statement }, "return", Expression, ";", "}"; Type = "int", "[", "]" | "boolean" | "int" | Identifier /\* one or more letters, digits, and underscores, starting with a letter \*/ ; /\* assignment statement \*/ Statement = "{", { Statement }, "}" | "if", "(", Expression, ")", Statement, "else", Statement | "while", "(", Expression, ")", Statement | "System.out.println", "(" , Expression, ")", ";" | Identifier, "=", Expression, ";" | Identifier, "[", Expression, "]", "=", Expression, ";" ; /\* boolean or matheamtical model expression\*/ Expression = Expression , ( "&&" | "<" | "+" | "-" | "\*" ) , Expression | Expression, "[", Expression, "]" | Expression, ".", "length" | Expression, ".", Identifier, "(", [ Expression { ",", Expression } ], ")" | IntegerLiteral /\* one or more decimal digits\*/ | "true" | "false" | Identifier | "this" | "new", "int", "[", Expression, "]" | "new", Identifier ,"(" ,")" | "!", Expression | "(", Expression, ")" ; <keyword> ::= abstract | boolean | break | byte | case | catch | char | class | const | continue | default | do | double | else | extends | final | finally | float | for | goto | if | implements | import | instanceof | int | interface | long | native | new | package | private | protected | public | return | short | static | super | switch | synchronized | this | throw | throws | transient| try | void | volatile | while

**3. Question 3, posted in GitHub**

**4. Question 4 Answer below**

The form "while B do S end" can be written as:

{ P } **while** B **do** S **end** { Q }

where P is the Precondition and Q the postcondition

The four criteria to prove while loop construct correct are:

P => I

{I and B} S {I}

(I and (not B)) => Q

the loop terminates

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

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

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

Criteria 4: the loop terminates

**Proving Correctness of given while loop:**

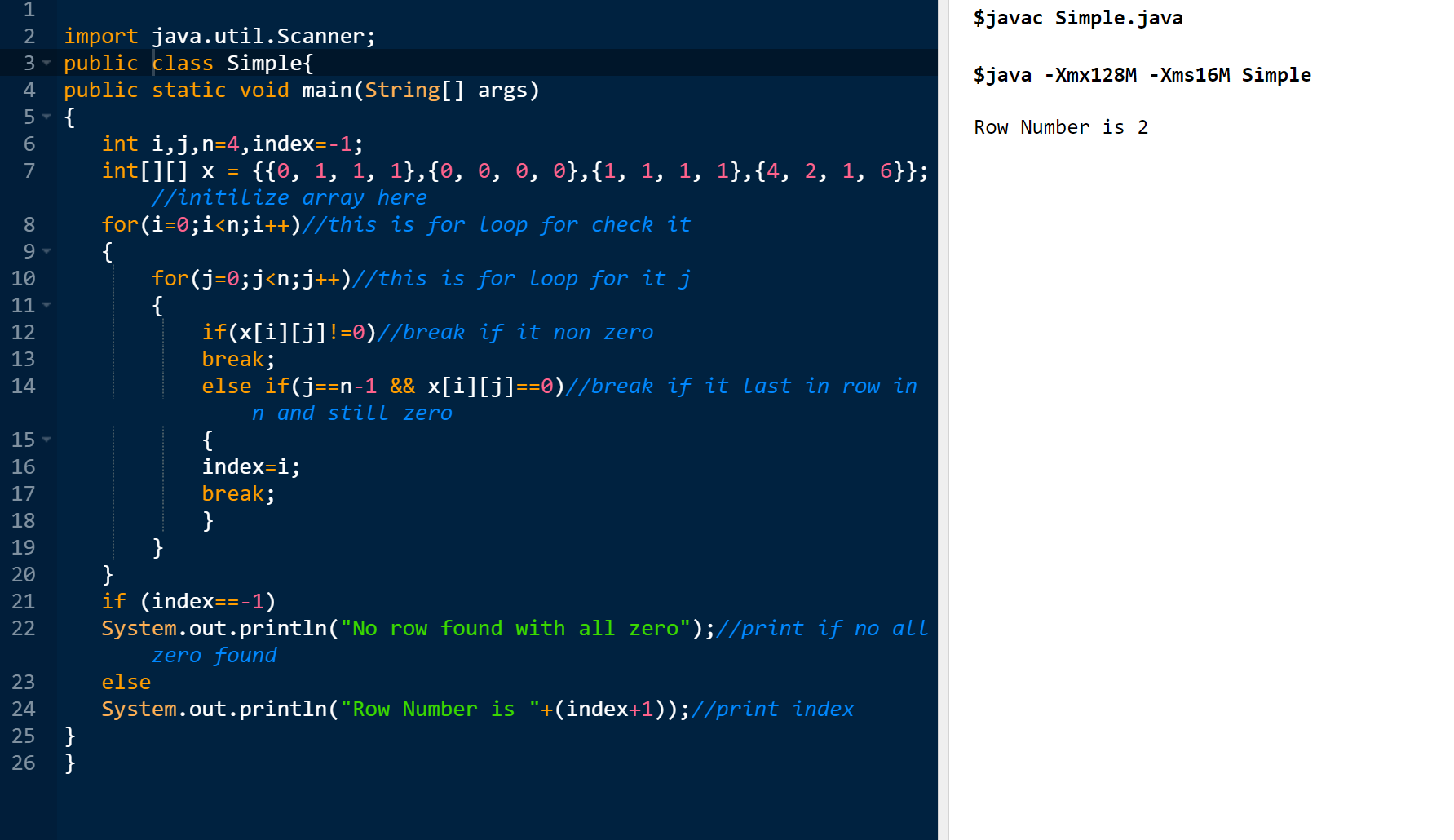
Criteria 1: The invariant that can be used here is that Power>0 . This holds true initially.

Criteria 2: After loop is executed value of power is still more than 0.

Criteria 3: When i>n i.e. B gets false but I is true, the Q is true. After the loop power = x^n which is same as Q

Criteria 4: After B gets false loop terminates.

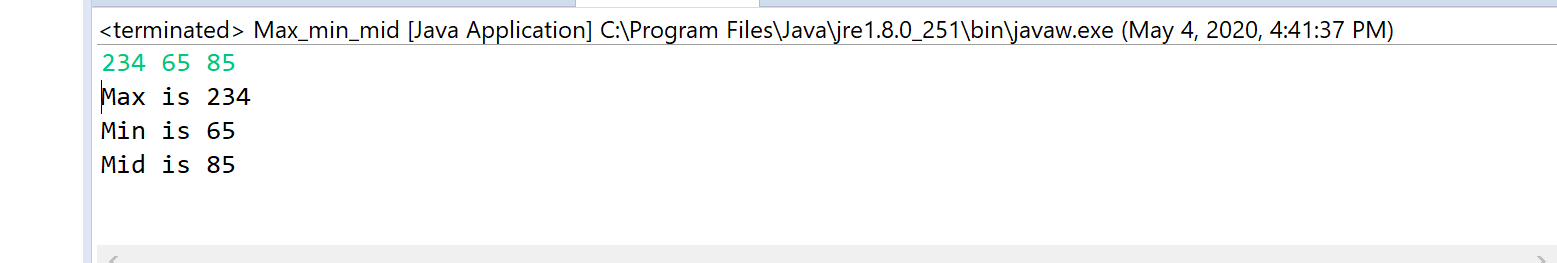
**5.Question 5. Codes in GitHub**



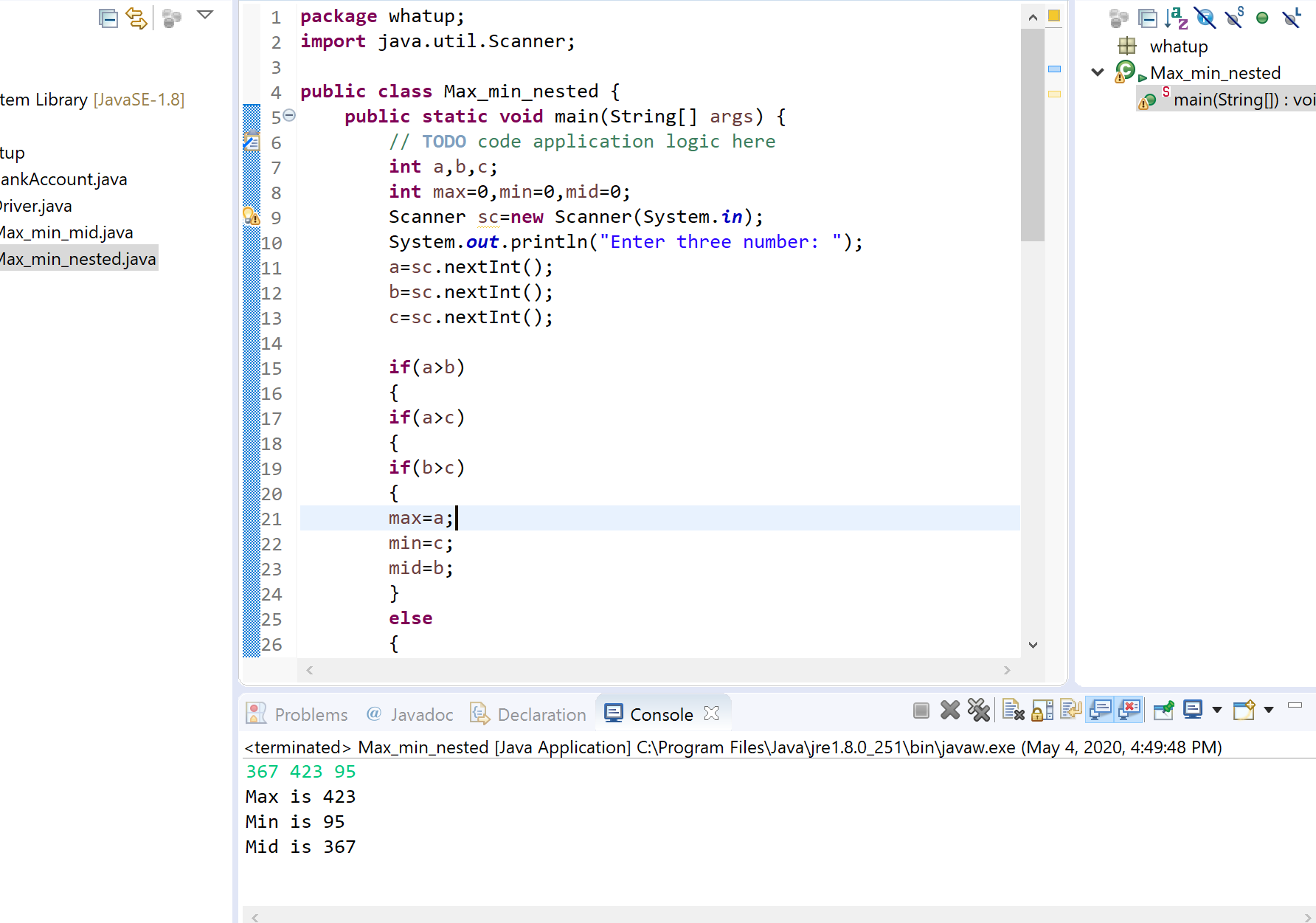
**Explanation:** In this case when it comes to comparing readability, the code with the Goto Statement compared to my statement without Goto, the no of times the loops runs to find the first row with all zero values is less than my without Goto Statement. Because of this result it is efficient to use Goto that was giving to us in the example.

**6.QUESTION 6 Both Codes in GitHub**

**6A. Code without nested selections screenshot**



**6B:Code with nested selections screenshot**



**Explanation for question 6:** The code with the nesting selection is more complex and the readability compared to the code without the nested selection. Also the code without the nested selection is Hard to read since there is multiple loops nesting together, which will make it hard to track and easy to get lost. Speed wise code with nested selection is much better than code without nested selection because in code without nested selection , each time it executes it will check each condition because we have not used any else statements. But in the code with nested selection, there will be chances where code can skip multiple conditions which makes this code more reliable and faster.

**7. Question 7.**

|  |
| --- |
|  |
| Dangling pointers, memory leaks are the memory access errors which are a serious program bug. The dynamic semantic checks for the memory access errors are created such that checks are fast and never generate false solutions. Here tombstone and lock and key are types of dynamic semantic checks. |
|  |
| A tombstone method: It is used to catch the memory access errors to objects through stacks and heap. Tombstone is allocated whenever an object is allocated in a heap or in a stack .Here the pointer consist of the address of the tombstone ,where tombstone consist of the address of the object ,when the object is claim back ,then the modification of tombstone is done such that tombstone contains a value (zero) which is invalid address. Tombstone are created as static objects in order to avoid important cases in the created code. When the program calls the deallocation operation heap objects can easily nullify tombstone. Hence to validate a tombstone ,it's better to link all stack objects together in a list which are sorted by the address of the stack frame in which the object lies. |
|  |
| Hence tombstones can be expensive, both in time and in space. |
|  |
| The time overhead consist of :- |
|  |
| (1) Allocation: When allocating heap objects or using a “pointer to” operator during making of tombstones |
|  |
| (2) Valid Check: It checks for validity on every access. |
|  |
| (3) Double indirection: It is time consuming. As the location of an object present in the heap changes quickly. |
|  |
| The space overhead consist of :- |
|  |
| 1. Size of Tombstone : A long-running program that creates and claim back objects continuously will ultimately run out of space for tombstones .Although the size of tombstone is less than the object to which it refers but it's creation and claiming back continuously will increase the space overhead. |
|  |
| 1. Due to tombstones space and time overhead, they are used less often in language implementations. They are used in Macintosh operating system. |
|  |
| 1. Locks and Keys: An alternative to tombstones is Locks and keys. A probabilistic protection from memory access errors is provided as it works for objects in the heap .Lock and key method is simple as it provides an address and a key to every pointer in a tuple .Each object in the heap has a lock in the beginning .When the key in the pointer is same as that of lock in the object when the pointer to an object in the heap is considered valid.A new key value is created whenever a new heap object is allocated at run time. |
|  |
| 1. In case of lock and key method when an object is claimed back, its lock is changed to some arbitrary value (e.g., zero) so that the keys in any remaining pointers does not match. If the block is continuously reused for another purpose, then the location that used to contain the lock will be restored to its former value by coincidence will be different . |
|  |
| 1. Overhead of locks and keys in terms of time and space are :. |
|  |
| 1. Extra space: Storage is increased as it adds an extra word of storage to every pointer and to every block in the heap. |
|  |
| 1. Copy: As it copies one pointer into another so cost increases .It compares locks and keys on each access that also adds to extra cost. |
|  |
| Pascal compilers are implemented by lock and keys method. |
|  |
| Therefore, in terms of implementation cost it is tough to say which is more cheaper as a tombstone check may result in two cache misses (one for the tombstone and one for the object)and lock and key copies one pointer into another so cost increases .It compares locks and keys on each access that also adds to extra cost. |
|  |
| In terms of safety lock and key method is more secure as it provides the probabilistic protection as compared to tombstone method.  **Question 8:**  **PART A)**  j = -3;  for( i = 0 ; i < 3 && j < =0 ; i++ )  {      switch( j + 2 )      {          case 3 :          case 2 : j--;                   continue;          case 0 : j += 2;                   continue;          default : j = 0;      }        j **=** 3 **-** i**;**  **}**  **PART (B)**  j = -3;  for( i = 0 ; i < 3 ; i++ )  {      if( j + 2 == 2 || j + 2 == 3 )          j--;      else if( j + 2 == 0 )          j += 2;      else          j = 0;        if( j > 0 )          goto A        j = 3 - i;  }  // create a new label  A **:**  **PART (C)**  j = -3;  for( i = 0 ; i < 3 ; i++ )  {      if( j + 2 == 2 || j + 2 == 3 )          j--;      else if( j + 2 == 0 )          j += 2;      else          j = 0;        if( j > 0 )          break;        j = 3 - i;  }  **PART (D)**  Inside the for loop, we enter the switch statement. We match the value of j + 2 in the different cases in the switch block  If j + 2 == 3 , then the control goes to the case when j + 2 == 2, then the j-- is executed.  If j + 2 == 2 ,then the j-- is executed.  If j + 2 == 0 ,then the j += 2 is executed.  The switch statement is completed, and the if block is executed and if j is greater than 0, then we go out of loop. If j is negative, then j = 3-i is executed and the loop is executed again.  **9.Question 9 Comparing JAVA to 2011 C++**  **Explanation:** C++ and JAVA both are object oriented programming language but C++ is platform dependent but JAVA is platform independent. By using C++ we can easily develop system programming but using JAVA we can be used to designed application programming very easily. We can differentiate C++ and JAVA in different aspects such as  KEYWORDS  Keywords are the reserve words in any programming language which having a specific meaning of its own.  Now the latest C++ supports 73 keywords but JAVA supports 52 keywords. Some of keywords are common to both the language, some of them are this,throw,switch.volatile etc…..  Some exclusive JAVA keywords are super,throws,synchronized etc….  Some exclusive C++ keywords are template,typedef,struct,virtual etc….  DATA TYPES            Data types are the keywords which are used for declaration of variables or objects. C++ and JAVA provides their own data types for users.  C++ supports data types as  Basic data types : char,int,float,double  Derived Data types : array,pointer  User defined data type : struct,union,enum,class  With basic data types we can also use modifiers like short,long,signed,unsigned such as long signed int, long unsigned int etc….  JAVA supports data types as  Primitive data type ad non primitive data types.  Primitive data types includes : Boolean,char,byte,short,int,long,float,double  Non primitive data types includes : Strine,Array,Class etc…            In JAVA the char data type occupies 2-bytes to store international character sets where as in C++ it occupies 1 byte of memory.  CONTROL STRUCTURES  All most all languages supports different types of control structures to manage the flow of control of statements in a program. Basically the control structures are  1. Sequential flow  2. Conditional flow  3. Iterative flow  4. Jumping statements  As the name specifies in sequential flow the statements will execute sequentially as they mentioned in the program.  In conditional flow depending on a certain condition few statements will execute and few will left.  In iterative flow a statement(s) will execute repeated number of times depending on the condition.  In C++ if,if else,switch are used for conditional flow and while,do…while and for is used for iterative flow and break,continue and goto are used as jump statement.  In JAVA if,if else,switch are used for conditional flow and while,do…while and for is used for iterative flow and break,continue are used as jump statement.  Expressions            An expression can be defined as a collection of operators and operands.  Depending on the types of data types used in programming languages the expressions can be categorized as  **Unary Expression :** A expression contain one operand and a unary operator.  **Binary Expression :** An expression uses two operands and one operator.  **Ternary expression :** An expression uses three operands separated by ternary operators.  If any expression consists combined operations such as unary,binary and ternary then it is called as combinational expressions.  If an expression consists of assignment statement then it is called as assignment expression.  **Logic**  If an expression consists of logical operators then that will be called as logical expression.  **Order or operations**            Both C++ and JAVA supports all these types of expressions. Java supports >>>(shift) and instanceof operator where as c++ does not support these two. C++ supports scope resolution operator but JAVA does not supports this.  The execution of expressions will depends on the precedence of operators. |

