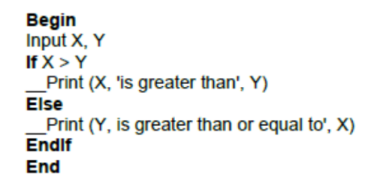
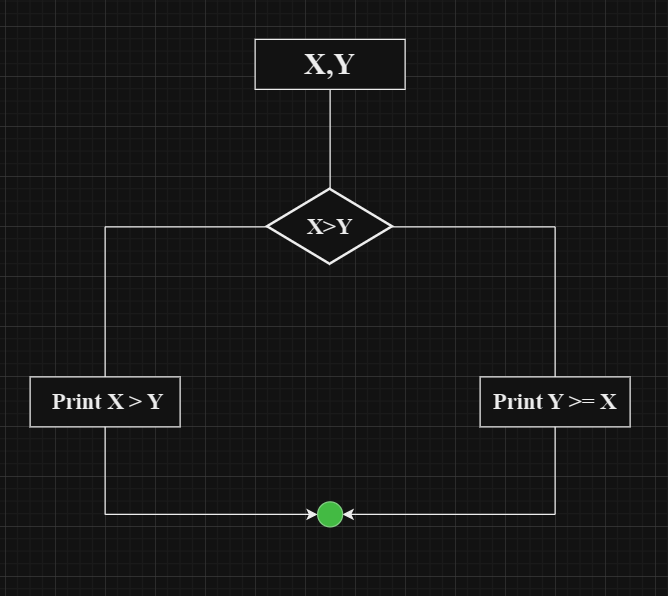
### **Sarcina 1.**

**Afișează pseudocodul de mai jos în formă de schemă logică.**





**Care este numărul minim de cazuri de testare necesare pentru a garanta o acoperire de 100% statement și 100% decision?**

A. Statement coverage = 3, Decision coverage = 3

B. Statement coverage = 2, Decision coverage = 2

C. Statement coverage = 1, Decision coverage = 2

D. Statement coverage = 2, Decision coverage = 1

**B.) Statement coverage =2, Decision coverage = 2**

Codul are două ramuri (branches): una pentru instrucţiunea ‘‘IF‘‘ şi alta pentru instrucţiunea ‘‘ELSE‘‘. Pentru a obţine acoperirea ramurilor trebuie să testăm ambele ramuri. Prin urmare , avem nevoie de 2 teste pentru acoprirea ramurilor.

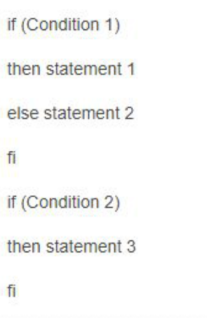
În plus codul are 3 instrucţiuni: pentru instrucţiunea IF, instrucţiunea de atribuire ‘‘Print X>Y‘‘, pentru instrucţiunea ‚‘‘ELSE‘‘ avem instrucţiunea de atribuire

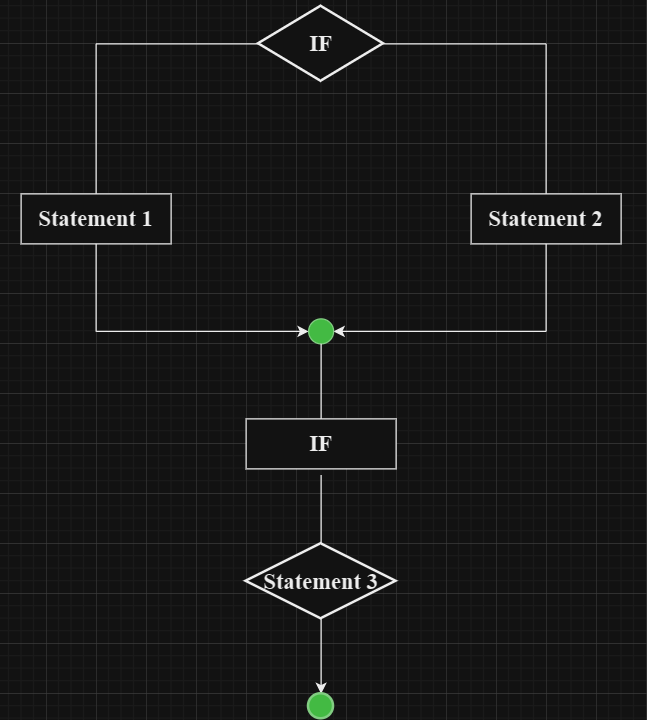
‘‘Print Y>=X‘‘, şi instrucţiunea ‘‘Print‘‘.

Pentru a obţine acoperirea instrucţiunilor trebuie să testăm toate cele 3 instrucţiuni. Prin urmare avem nevoie de 2 teste pentru acoperirea instrucţiunilor.

### **Sarcina 2.**

### **Afișează pseudocodul de mai jos în formă de schemă logică.**





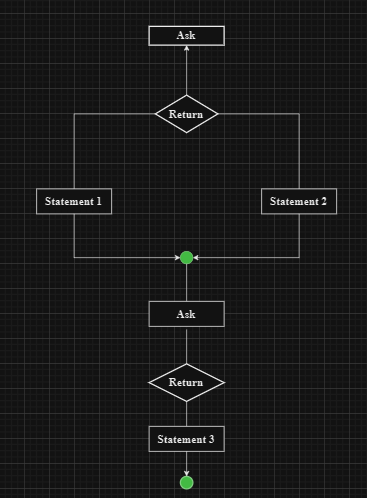
**Care este numărul minim de cazuri de testare necesare pentru a garanta o acoperire de 100% path?**

A. 1

B. 2

C. 3

D. Niciun răspuns nu este corect



Pe baza diagramei de sus avem următorul tabel:

|  |  |  |
| --- | --- | --- |
| **Test case** | **Question 1** | **Question 2** |
| 1 | Statement 1 | Statement 2 |
| 2 | Statement 3 | - |

Pe baza tabelului de mai sus minim 2 teste sunt necesare pentru a garanta o acopeire de 100 % path.

Răspuns corect **B.) 2**

**Sarcina 3.**

**În cadrul unei echipe care dezvoltă o aplicație Java, clarificați cine din echipă este responsabil pentru teste white-box/static și ce tool-uri sunt utilizate în acest scop?**

**Pentru această sarcină, este necesar să efectuați un research de sine stătător și să căutați exemple de tool-uri și framework-uri folosite pentru testarea aplicației scrisă în Java.**

Java oferă o gamă largă de instrumente pentru diferite părți ale procesului de dezvoltare, de la codificare până la lansarea aplicațiilor. Unele instrumente sunt incluse în Java Development Kit (JDK), în timp ce altele sunt instrumente terțe dezvoltate de comunitate sau de alte companii. Java are o varietate de instrumente pentru a îmbunătăți eficiența aplicației, calitatea codului și productivitatea generală a aplicației. Mai jos sunt instrumentele Java care pot fi extrem de utile pentru dezvoltatorii Java:

**Integrated Development Environments**

*Eclipse*-Un mediu de dezvoltare integrat (IDE) cu sursă deschisă popular, cu pluginuri vaste și suport comunitar, acestea sunt utilizate în principal pentru dezvoltarea Java.

*Intellij IDEA-*este un mediu de dezvoltare integrat (IDE) ce acceptă o gamă largă de limbaje de programare, este deosebit de favorizată în comunitatea Java.

**Build Tools and Dependency Management**

*Maven -*este atât un instrument de compilare, cât și un instrument de gestionare a dependențelor.

*Gradle-*procesează construcția printr-un ciclu de viață care include sarcini precum inițializarea, configurarea și execuția.

*Dependency Management*

**Version Control System**

*Git*-Git este în prezent cel mai popular VCS, iar platforme precum GitHub, GitLab și Bitbucket sunt construite în jurul lui Git. Este folosit în principal pentru a urmări modificările codului sursă în timpul dezvoltării software, permițând mai multor dezvoltatori să colaboreze la un proiect.

**Continuous Integration/Continuous Deployment (CI/CD)**

*Jenkins*-automatizează construirea, testarea și implementarea aplicațiilor.

*Travis CI-*poate rula automat teste și raporta dacă acestea trec și vă arată că arborele dvs. de lucru este curat, fără nimic de împins.

**Testing Tools**

*JUnit* -este folosit pentru scrierea și executarea testelor unitare în Java.

*Mockito -*este deosebit de util pentru crearea de obiecte simulate în testele unitare.

**Static Code Analysis Tools**

*SonarQube- v*erifică standardele de codare pentru aplicație, rafinarea codului, claritatea codului, potențialele erori și vulnerabilitățile de securitate.