Q.1:- In EP testing, input domain of a program is partitioned into a finite number of equivalence classes such that one can reasonably assume, but not be absolutely sure that the test of a representative value of each class is equivalent to a test of any other value. Two steps are required to implement EP:

(a) The equivalence classes are identified by taking each input condition and partitioning into valid and invalid classes.

(b) Generate the test cases using the equivalence classes identified in the previous step.

But in case of BVA, test cases that are close to boundary conditions have a higher chance of detecting errors. Here boundary conditions mean an input value may be on the boundary or just above the boundary.

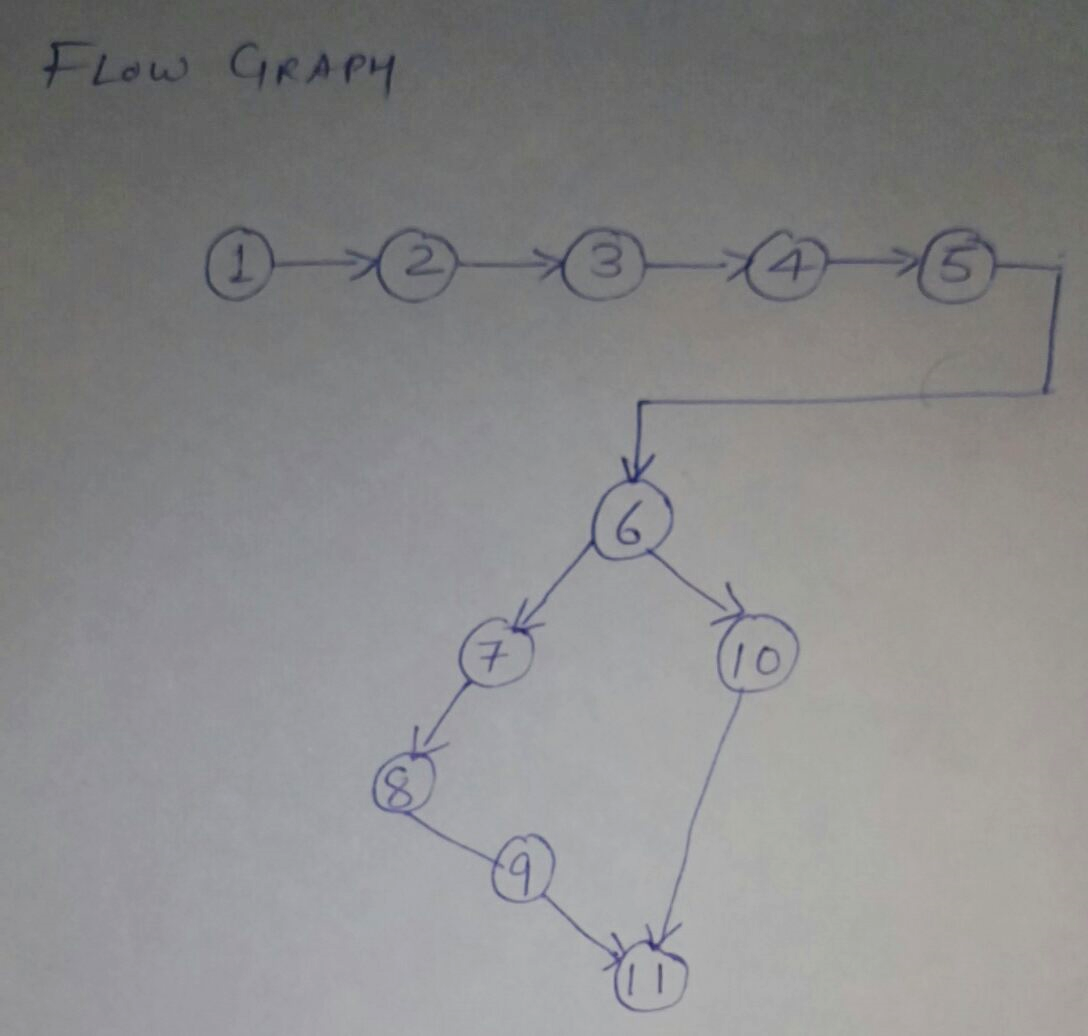
Q.2:- The control flow of a program can be analyzed using a graphical representation known as flow graph. The flow graph is a directed graph in which nodes are either entire statement or fragments of a statement and edges represent the flow of control.

A flow graph can be easily generated from the code of any problem. The basic construct of flow graph is given below: -

1. main()
2. {
3. int a,b;
4. printf(“Enter two numbers a & b”);
5. scanf(“%d %d”,a,b);
6. If(a>b)
7. {
8. printf(“A is greater”);
9. }
10. else printf(“B is greater”);

11. }

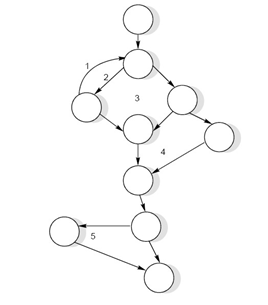
Flow graph:-



Q3. The Product Metrics describe the characteristics of the product. There are a number of parameters related to it:

1. Size
2. Complexity
3. design features
4. performance
5. efficiency
6. reliability
7. portability

Q4. Consider the following graph and calculate cyclomatic complexity by all 3 methods



No. of edges=13

No. of nodes=10

V(G)=e-n+2p =13-10+2(1) =5

V(G)=+1=4+1=5

V(G)=no. of regions=5

Q.5 Consider the McCall quality model. The attributes for software quality analysis:-

|  |  |  |
| --- | --- | --- |
| 1 | Reliability | The extent to which a software performs its intended functions without failure. |
| 2 | Correctness | The extent to which a software meets its specifications. |
| 3 | Consistency & precision | The extent to which a software is consistent and give results with precision. |
| 4 | Robustness | The extent to which a software tolerates the unexpected problems. |
| 5 | Simplicity | The extent to which a software is simple in its operations. |
| 6 | Traceability | The extent to which an error is traceable in order to fix it. |
| 7 | Usability | The extent of effort required to learn, operate and understand the functions of the software |

|  |  |  |
| --- | --- | --- |
| 8 | Accuracy | Meeting specifications with precision. |
| 9 | Clarity & Accuracy of documentation | The extent to which documents are clearly & accurately written. |
| 10 | Conformity of operational environment | The extent to which a software is in conformity of operational environment. |
| 11 | Completeness | The extent to which a software has specified functions. |
| 12 | Efficiency | The amount of computing resources and code required by software to perform a function. |
| 13 | Testability | The effort required to test a software to ensure that it performs its intended functions. |
| 14 | Maintainability | The effort required to locate and fix an error during maintenance phase. |
| 15 | Modularity | It is the extent of ease to implement, test, debug and maintain the software. |
| 16 | Readability | The extent to which a software is readable in order to understand. |
| 17 | Adaptability | The extent to which a software is adaptable to new platforms & technologies. |
| 18 | Modifiability | The effort required to modify a software during maintenance phase. |
| 19 | Expandability | The extent to which a software is expandable without undesirable side effects. |
| 20 | Portability | The effort required to transfer a program from one platform to another platform. |