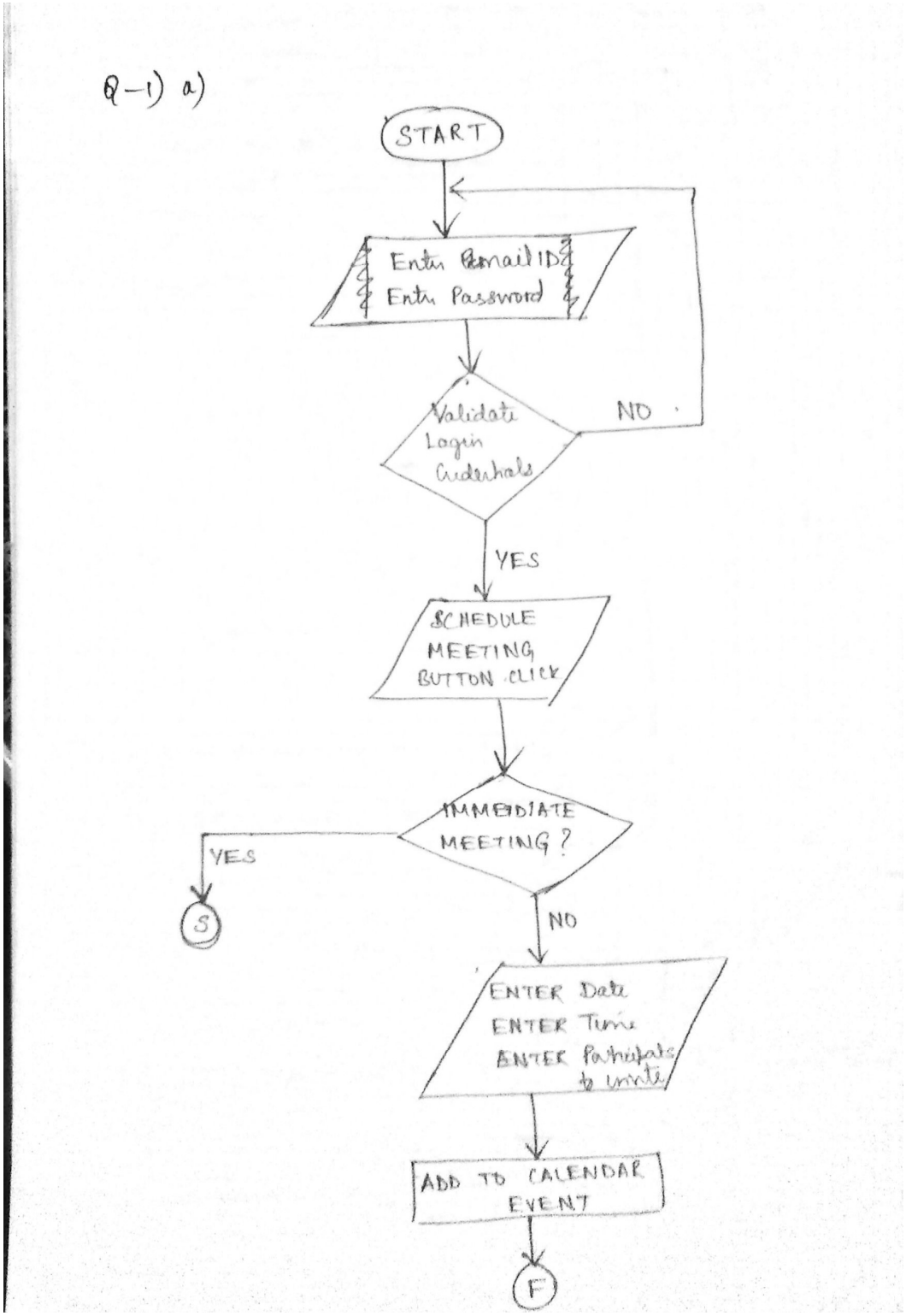
PART B

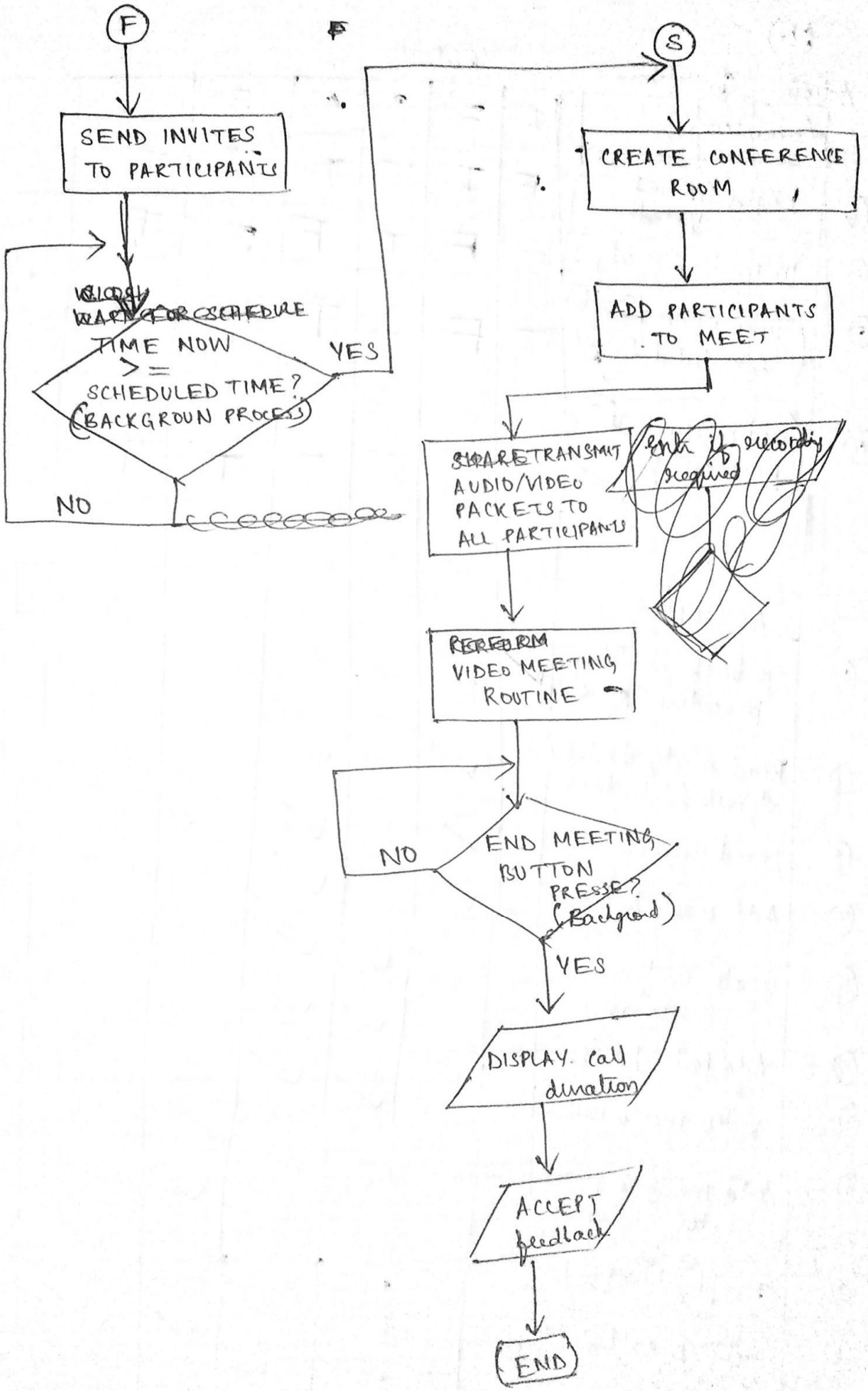
1. Consider Google Meet Application and a scenario of scheduling a meeting.

a. Design the component for this scenario using a flow chart (4 Marks)

b. List the actions, conditions and rules in this scenario and represent them using a decision table (3 Marks)

c. Represent the design of the component in Pseudocode. (3 Marks)
List your assumptions about the scenario clearly.




F
SEND INVITES TO PARTICIPANTS
TIME NOW >= SCHEDULED TIME?
(BACKGROUN PROCESS)
YES
NO
S
CREATE CONFERENCE ROOM
ADD PARTICIPANTS TO MEET
TRANSMIT AUDIO/VIDEO PACKETS TO ALL PARTICIPANTS
VIDEO MEETING ROUTINE
END MEETING BUTTON PRESSE? (Backgrond)
NO
YES
DISPLAY call duration
ACCEPT feedback
END

(b).

Action / Condition	Rules						
Conditions	1	2	3	4	5		
① Login ~~Correct~~ Valid.	F	T	T	T	T	T	
② Immediate meety ?	–	F	T	F	F	T	
③ (Time now ~~is after~~ scheduled time ?)	–	F	–	T	T	–	
④ (End meety button pressed ?)	–	–	–	–	T	T	
ACTIONS							
① Go back to "login" procedure ~~proceed~~	✓	1					
② Read meety shedul details (data, tim)		✓		✓			
③ Send invites		✓		✓			
④ Add meety to calndr		✓		✓			
⑤ Create conferuce room			✓	✓			
⑥ Add Participants			✓	✓			
⑦ Video meety routn			✓	✓			
⑧ End meety in progres (close room)					✓	✓	
⑨ accept feedback					✓	✓	
⑩ display call dtails					✓	✓	

(c) Pseudo Code

```
BEGIN

login:    READ email ID
          READ password
          IF (user credentials are invalid) THEN
              GOTO login
          END IF

          READ schedule meeting immediate (or) later
          IF (NOT immediate meeting) THEN
              READ "date"
              READ "time"
              READ "participants" to invite

              ADD_TO_CALENDAR (date, time, meet ID generated)
              SEND invitation to "participants".
              FOR (EACH p in participant) DO
                  SEND INVITE to p
              END FOR.
              IF (current date time >= schedule date, time) THEN
                  GOTO meet_start
              END IF
          END IF

meet_start: Create conference room
          ADD all participants

meet_prog: PERFORM VIDEO MEETING ROUTINE
          IF ("end-meet" button NOT pressed) THEN
              GOTO meet_prog
          END IF
          DISPLAY call duration
          READ feedback

END.
```

Assumptions

1. User must be logged in.
2. User can schedule immediate / future meety
3. User can end meety at any time, if there is a room in activation
4. User can view the meety statistics & give feedback

2.

a) How do coding standards for a one-person software production company differ from those in organizations with 300 software professionals?

b) Why is regression testing an important part of any integration testing procedure?

c) Differentiate white box and black box testing methods.

q-2) (a) Under ideal circumstances, there should be NO existy defference in the coding standards followd in a one-person software production from that of having 300-professionals in a team

This is because, a one-person production, can at any time expand into a large organisation if they meet success and good revenue after the initial release.

Even though, the testing and quality assurance is performed by the same person, & he is likely to know everything about his software without the need to follow coding standards, it is very important to adopt them.

— If the company expands, dedicated SQA teams may be appointed. They will find it hard to test & maintain the codebase.

— If the software is sold, the buyer will have to invest a lot of time in comprehending the code structure and modules

— Even if the same person, wants to edit/maintain the codebase post delivery after a long time, he has to rely on his memory & may get confused if good naming conventions are not used, and modules are not well organised, don't have ideal coupling & cohesion, etc.

Moreover, some standards like good "separation of concerns", non-ambiguous and simple code are beneficial to the performance, & reliability and robustness of the artifacts. This aspect, is independent of the development team size & must be always followed.

(b) Regression testing involves re-running a subset of already executed test-case to ensure that changes have not propagated or "side-effected" the rest of the software modules / artifacts.

As in Integration testing, whether top-down, bottom-up (or) sandwiched, always involves frequent changes at different extents of integration (with stubs or actual modules). As & when bugs are detected, they have to be debugged, by adopting a suitable strategy (brute force, backtracking, induction, etc) to find the source of error

In all cases, modules undergo changes & some aspects are modified.

The dependent modules ~~will hence be~~ are very likely to be impacted by such modifications on the procedure or data are made.

A regression test after every such update ensures that if a bug has been induced or has caused unintended behaviour, it can be identified and corrected early-on

Hence it corrects such unforeseen effects on other modules

This may common, especially in case of "rapid cycle tests" when updates are made ~~onto~~ very frequently ...

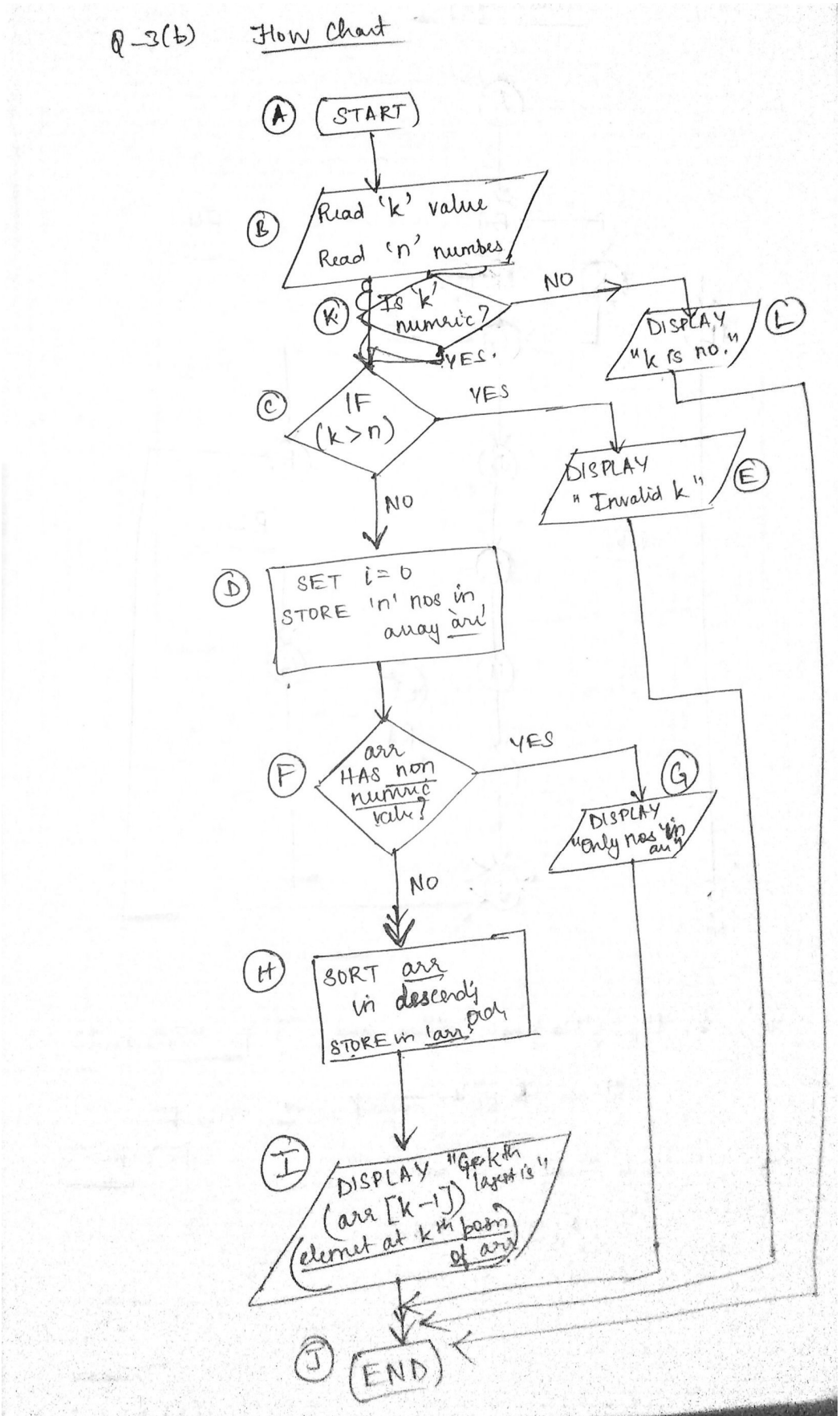
(c)

White Box Testing	Black-Box Testing
* Internal structure is exposed to the tester & is the focus area of testing	Internal structure of the software/ program is hidden.
* Implementation knowledge is essential.	Implementation knowledge is not required.
* It is a structural ~~test~~ of (behavioral) test of software	It is a functional test of and logical software.
* A very time-consuming process	Not as time consuming as white-box testing.
* Common techniques include → Basis Path testing ← (analysis of input flows) → Control structure Testing ↳ data flow ↳ loops ↳ conditions testing.	Common techniques include → Equivalence Partioning (divide i/p values into classes) → Boundary value analysis →

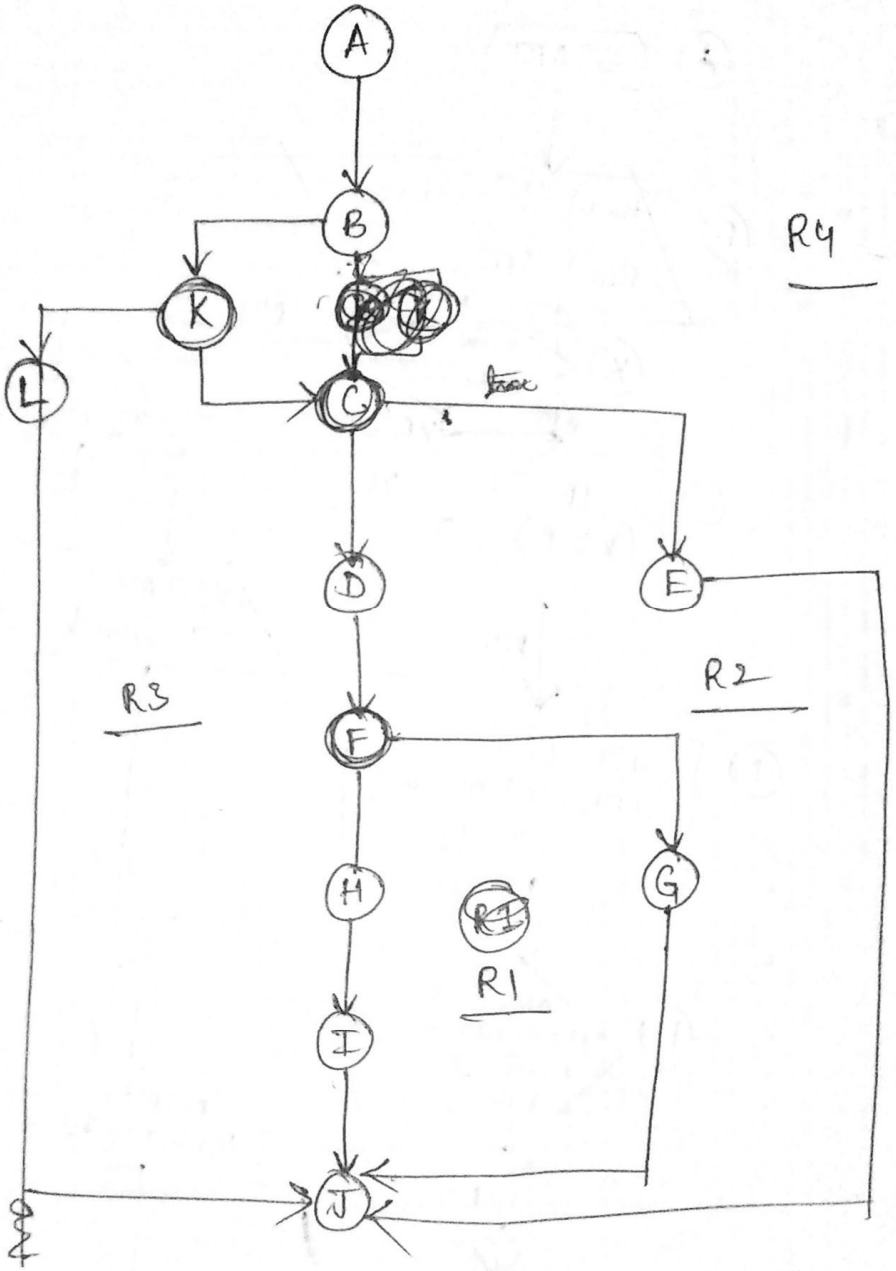
PART C

b) A program finds the kth greatest of the given numbers. Develop a set of test cases that you feel will adequately test this program using basis path testing method.

a. Draw the control flow chart (2 Marks)
b. Represent it as a flow graph (2 Marks)
c. Find the number of independent paths (2 Marks)
d. Develop the test cases (4 Marks)



ii) Flow Graph



iii) In the flow graph, there are 3 predicate nodes

(ie) K, C and F

(ie) P=3

∴ No of independent paths

$V(G) = P + 1$

$= 3 + 1$

$= 4$

graph also has 4 regions R1 to R4

d) Test Cases

(TC)	Inputs	Expected O/P	Expected Flow
(1)	k = "hello" arr = [1, 2, 3, 4, 6] (i.e) n = 5	k is no.	A → B → K → L → J.
(2)	k = 100 arr = [5, 3, 1, 2] (i.e) n = 4	Invalid k. (k < n).	A → B → K → C → E → J.
(3)	k = 2 arr = ['a', 10, 'b', 11] (i.e) n = 4	Only nos in array.	A → B → K → C → D → F → G → J
(4)	k = 3 arr = [10, 9, 3, 11] (i.e) n = 4	k^{th} layest is 9	A → B → K → C → D → F → H → I → J

All the 4 independent paths are covered through the test cases.