Introduction To Computer Problem Solving

* Introduction to Computer Problem Solving

* 1. Programs and Algorithms:-

The computer Solution to a problem each set of explicite and Unambiguous Instructions express in programming Language, This set of Instructions is called program.

An Algorithm is step by step representation of problem Solving steps. An algorithm Correspond to a Solution to a problem that is Independant of any programming Language.

To obtain the Computer Solution to a problem we have to supply the Program with input or data. The program than take this input and manupulate it according to it's instructions & produces an output which represents the Computer Solution to the problem.

2 Requirements For Solving by Computer:-

After Studying even a small sample of Computer problem it soon becomes obvious that the Conscious depth of Understanding is needed to design effective Computer algorithm.

3. Problem Solving Aspect :-

It is widely recognize that problem Solving is a Creative process which Largely defines Systematization and mechanization. There are verious ways and steps about the understanding and Solution of the problem.

1.1 Problem defination phase :-
(Understanding)

Success in solving any problem is possible only when understand the problem at hand. Make useful progress in solving a problem by preliminary Investigation which will be taught as problem defination phase. During this phase workout what must be Done rather than How to do it that is we must try to extrack From the problem statement, set of precisely define task. The development of algorithm For Finding square root or greatest Common deviser (GCd) are good example of how important it is to carefully define the problem.

1.2. Getting Started on a problem :-

There are many ways to Solve most problems and also many Solutions to most solution and this situation does not make job of problem Solving easy. We get Conflict between which part ~~a~~ are Likely to be Fruitless and which part are productive.

In such situation the best advice is not to be to Concerned about detail. The details Come Later when the Complexcity of that problem brought Under Control.

1.3 The use of specific example:-

A useful Stategy is to use Some props ~~to try~~ to get a Start with ~~the~~ problem with the help of Specific example.

1.4 Similarities among the problem:-

We have already seen that one way to make a start on a problem is by Considering specific example~~m~~. The ~~ert~~ another thing that we Should always try to bring as much past experiance as possible, To bear on Current problem. In this respect it is important to see. If there as any similarities between the Current problem and the other problem, hence it is very Important SKill that develop Problem solving from variety of angles once you develop a skill it is possible to get started on any problem.

1.5 Working backwards from the solution :-

We should write down as we go along the verious steps and explorations made which will help for the Investigation and avoid duplication of efforts. Once we have solved the problem, reflect back on the way we went about ~~des~~ discorering the solution. This Can help us significantly. In short we said that "We learn most when we have to Invent".

1-8-18

1.6. General problem solving strategies :-

There are no. of general & Powerful Computation strategies that are repitedly use in verious guises in Computing Science. The most widely Known & Commonly use strategy is divide & Conquer strategy in this strategy the Basic idea is ~~de~~ divide the original problem into two or more sub-problems. which can be solved more effeciently by the same technique. This type of solution has wide applica-

-tims in many more real life problem such as Sorting, Selection, Searching etc.

4. Top-DOWN Design :-

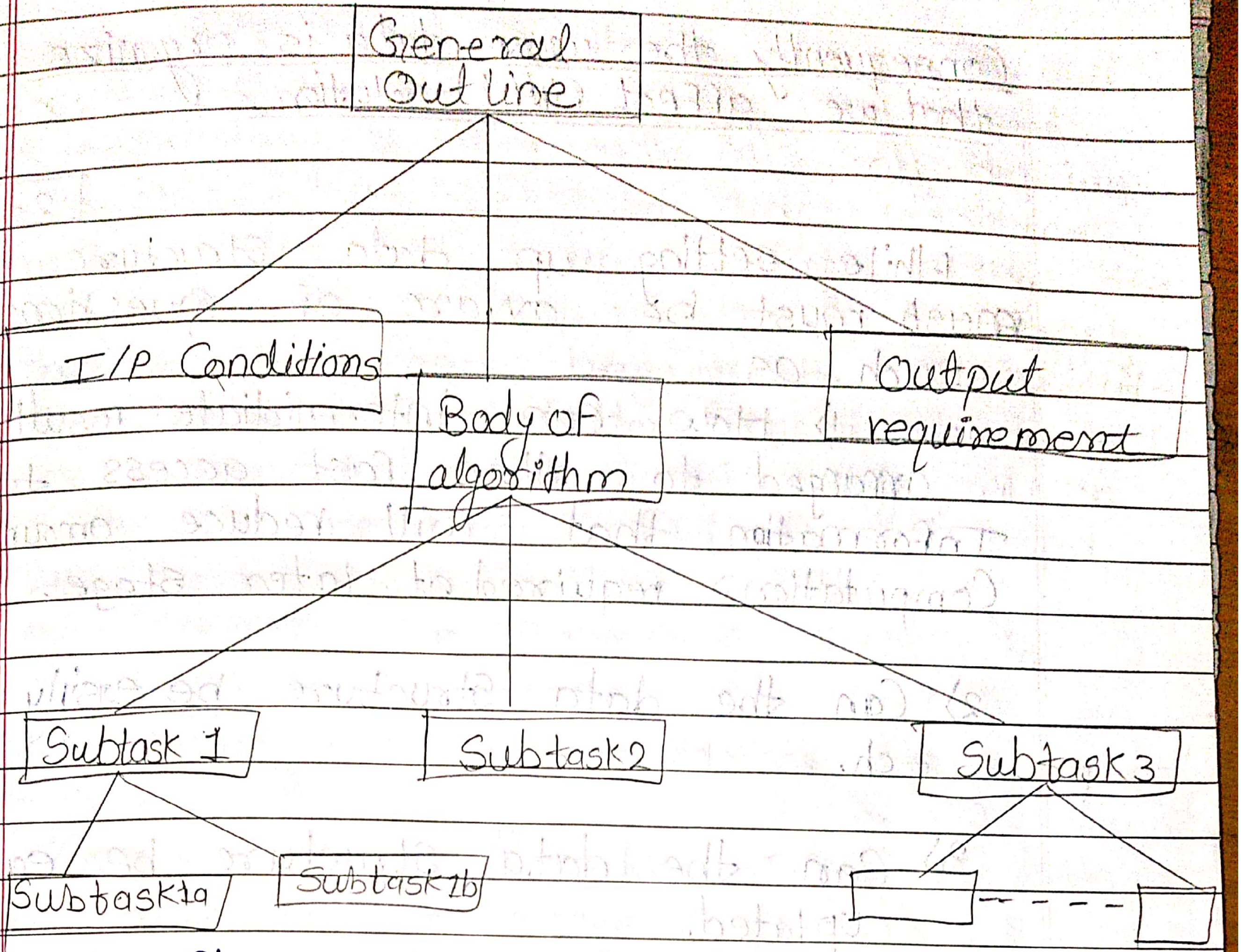
The primary goal in Computer problem Solving is an algorithm which is Capable of being implimented as Correct & efficient Computer Program.

A technique for algorithm design that tries to accommodate this human limitation is Known as top-Down design or Step wise refinement.

Top-Down design is a strategy that we can apply to take the Solution of a Computer problem from a ~~Vageu~~ g Vague outline to Precisely define algorithm & program implimentation.

Following are the Various steps in top-down design.

1) Breaking a problem into Sub problem.



The Figure shows breakdown of Problem in ~~subst~~ Subtasks. The process of breaking down the position into Subtask in the manner discribes results implimentable sets.

② Choice of Suitable data structure:-

One of the most important ~~disi~~ decision we have to make in formulating Computer Solution to problem is the Choice of approprеate data structure, all programs opperate on data &

Consequently the way data is organize that we affect final solution.

2-8-18

While setting up data structure we ~~Mush~~ must be aware of Questions such as

1) How can intermidiate result be arranged to allow fast access to Information that will reduce amount of Computation required at later stage.

2) Can the data structure be easyly search.

3) Can the data structure be easyly updated.

4) Does the data structure provide the way of recovering and ~~at~~ earlier state in the Computatio

5) Does the data structure involve the ~~excessive~~ (accessive) use of Storage etc.

* Construction of Loop :-

Loop is ~~at~~ iterative Construcs that are Conditionaly exicuted these

~~structions~~ structures (Loops) together with input output statements, Computable and assignment expressions make up the heart of program implimentation. to Construct any loop we must take into acount three things the Initial condition that need to apply before the Loop begins to ~~st~~ exicute, The invariant relation (that is increment or decrement) that must apply after each ~~at~~ iteration of Loop. & third the ~~cond~~ termination Condition.

1) Establishing Initial conditions for Loop :-

Consider Loop Variable say i & it's value in the range $0 \leq i \leq n$

For example to calculate sum of n number initialy we have to assign ~~to~~ $(i := 0)$

$i := 0$ → assignment symbol

$sum := 0$

the value of $i := 0$ and sum of 0 number is 0.

2) Finding Iterative Constructs :-

Once we have the Condition

debug - error काढणे

For solving problem the next step is Construct the ~~o~~ Iterative Value. i.e. we built the solution to the problem for $i := 0$ to get $i := 1$ for that use the expresion as $i := i+1$.

8-8-18

* ~~Introduction of Algorithm~~

3) x Implementation of Algorithm :-

The Implemention of an algorithm is mechanical process which should be properly designed. If it is properly designed The path of execution should Flow in Straight Line from top to bottom. Such design is much easyer to understand and debug. They ~~there~~ also easy to modify and update

1) use of procedure to emphesize modularity :-

modulalizasion of the program is helpful for Implimentation and readability of the main program. This practice allows ~~o~~ us to empliment set of Independent Procedure to perform Specific and welldefined task.

For eg:- In an algorithm it is required to

Sort a list of name , then a specific Independant procedure to be use for the sort.

For eg:- ~~prod~~ ~~procedure,sort~~

```
procedure sort ;
  begins
     ____
     ____

        write in ("sort called")
    end.
```

2> choice of variable name :-

The program become more meaningful and easier to ~~be~~ Understand ~~and~~ If choice of appropreate variable and Constant name.

For eg :- If you want to enter day of the week we have to use proper variable name as —

§- day -week
week - day
day of week
days etc.

A cleare defination of all variables and Constant at the start of each procedure can also be very helpful.
each variable should ~~be~~ only have one role in the given programme.

~~3)~~3) Documentation of program :-

Documentation is the information that the program presents to the user duering the ~~eq~~ exeicution face A good programming practice is always write a program with documentation so that they can be exeicuted and use by other people who are unfamilier with working and ——— Input ↑requirement of the program. This means Program must specify exactly what responses it require from the user.

4) ~~Debugging in~~

4) Debugging Program

To make the task of ~~detecting~~ detecting Logical errors and sjntaxs error ~~this~~ is debugging phase. For that it is nessesary to go ~~through~~each and every step of program and also write the addition Print statement In the program which helps in removing the error.

5) Program testing -

It is often not possible write a program that handle all input Conditions that may be supplied for a given problem. Although it is not always practical to implement program that Can handle all possible input Conditions.

program testing for various values & types of Values in nessecceryies for the smooth exicution of the program.

For that some ~~types~~ tipes are there. use fixe numeric constants for the program testing and than ~~generalize~~ generalize it.

Proper implimentation of algorithm minimize the number of errors.

* Program Verification :-

The Cost of development of Computing Software has become a major expense in the application of computers. Experience in working with computer sujest that more efforts and resorces ~~and~~ are spend in correcting the errors in the program. The best Suggestion is for that more care should be taken while creating the code & writing the Code at the time of program development.

- program verification reffers to the

application of mathematical proofs techniques to ~~ex~~ establish the results obtain by the exicution of program with orbitory input.

1) Computer Model for program exicution-

to perssive the this goal of program verification we must fully appreciate what happens when a program is exicuted under the infuence of given input Condition. There are various paths for exicution but for a given set of input condition only one of these path will be followed.

2) Input & output assersion:-

This ~~a~~ very first step that need to be taken in order to prove A program ~~to~~ correct or not, for that provide a Formal statement of specification ~~for~~ of the variables & Input & output assersion which can be express in logic notation.

$$x = (q * y + r) \wedge (r < y)$$

* Implication & symbolic execution;

To proceed with the verification procedure it is usually to set up ~~to~~ no. of ~~Intem~~ Intermidiate Verification Conditions between the Input & output assertion.

16 - 8 - 18

* The Analysis of Algorithm :-

There are usually many ways to solve any given problem there are offen sertain aspects which are based on practicle level. We are useually ~~Inte~~ Intrested in a solution which is écconomical in the use of Computing and humon resources Good Algorithm useually posseses the following ~~quantiti~~ qualities & capabilities.

1) They are simple but powerful & General solutions.

2) They can be easyly understood by others that is the implimentation is clear & ~~cosi~~ Consize without being tricky.

3) They can be easyly modified If nessessary.

4) They are Correct for clearly defined Situations

5) They are able to be understood on no. of Levels.

6) They are economical in the use of Computer time, Computer storage & peripherors.

7) They are well documented & to be use by others who do not have detail Knowledge of their inner ~~working~~ working.

8) They are not depend on being run on a perticular Computer.

9) They are able to use as a sub prosidure for other.

10) The solution is pleasing & satisfing to it's designer so that designer fill proud to have created.

20-8-18

* What is algorithm:-

Planing a program with involving it's Logic. The term algorithm refers to the Logic of ~~a~~ program. It is a step by Step discription of how to arrive at the Solution of given problem.

Algorithm is sequence of ~~Fraten~~ Instruction that when exicuted in the specified sequence gives the desired result.

It must ~~pos~~sseses following characteristics.

1) each instruction Should be precise & Unambiguoues.

2) each Instruction Should be exicuted in a Finite time.

3) No Instruction Should repite infinitely. this ensu that algorithm terminets ultimetly.

4) after exicuting the instruction the desired results are obtained.

* Sample Algorithm:-

eg:-

Q.1. We have mark memo of 80 Students with gread on it as First, second, third, & fail. Calculate & Print the total No. of Student passed in First gread.

→ 1. Initialise total First & total marksheet to zero.

2. Take the marks of Next Student.

3. Check the division if it is FIRST if no go to Step 5.

4. Add 1 to total First

5. Add 1 to total Mark Sheet.

6. ~~Is~~ If total marksheet = 50? if no go
to step 2.
7. Print total first.
8. Stop.

eg:- 2 There are 100 emploies in an organiseti-on. the organize distribute based on the there performance recorded. & this is given by grade of three catagi-ries A - outstanding, B - good, C - Average.

If Grade A - 100% Bonus. (Gross Salary)
If Grade B - 70% (G.S)
If Grade C - 20%.

*Algorithm → 1. Initialise total Bonus & total emp. to zero.
2. Take the appreacial or gread.
3. Check the Gross salery & gread.
4. if gread = "A" bonus = G.S.
5. If grade = "B" bonus = 0.7 * G.S.
6. If grade = "C" bonus = 0.2 * G.S.
7. total bonus = G.S + bonus.
8. Add 1 to total emp.
9. Print total bouns.
10. If total emp. = 100? if no go to Step 2
11. Stop.

* Flow Chart :-

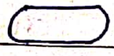
A Flow Chart is pictorial representation of an algorithm. It uses boxes of different shape to denote different types of Instructions. Programers Write actual Instructions with in these boxes using clear & consize statements

* Why use Flow chart?

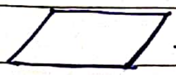
We First represent an algorithm as Flow Chart & then expess the flow chart in programing language to prepare Computer program.

The Advantage of drawing a Flow Chart is that, A Programer can Concentrate FUlly on the Logic of the solution of the problem without paying attension to the Syntex & other details of Programming Language. Because of flow Chart a programer can detect any error in the logic with greater ease than in the case of program. To reduce the no. of errors it is a good practise to draw a Flow Chart before writting a program.

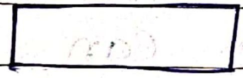
* Different Symbols of Flow Charts :-

1)  → terminal Symbol

It indicates beggining (start) & end(stop) in a program. It is First & Last symbol & only input & or output Flow Line is connected

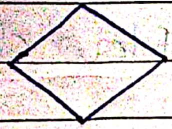
2)  → Input/Output

This symbol Indicates Input /output data from any type of Input/ Output devices & storages devices.

3)  → Processing

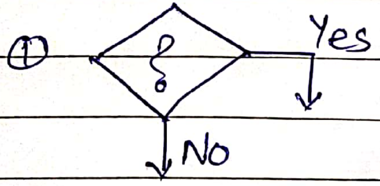
A processing symbol represent Arthematic & data moment Instructions all Arthematic processess ~~te~~ such as Addition, subtraction etc are indicated by this symbol.

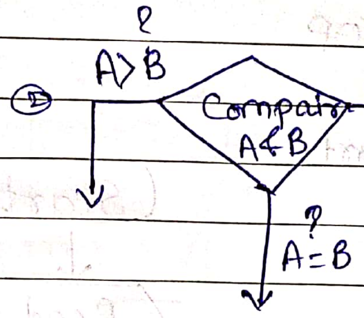
They also Indicate process of moving data from one mepmory location to unother (Such as Assignment Statement) This symbol has one Input & one output Flow Line.

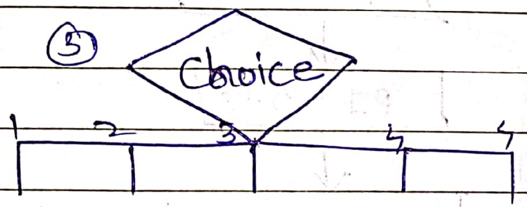
4)  DEcision Symbol

This Symbol Indicates a desision point which is a point at which branch to one of two or more alternative path is possible.

Three different types of Decision box are there



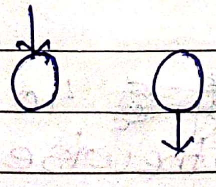




5* ○→Connector -

When a flow chart become so, Complex No. of flow Lines are confusing & When flow Charts spreads more than one page than the Connector are used Connect the flow Chart

It has either InPut line/ out-Put Line.
Flow

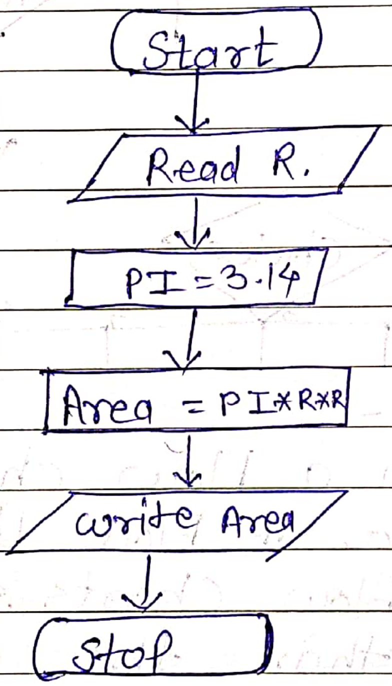


Date 27-8-18
Page

Q. Write an algorithm & draw a Flow chart For Calculating Area of Circle.

→ 1) Start
2) Read Radius R of Circle
3) Assign PI := 3.14
4) calculate area = PI * R * R
5) Write Area
6) STOP

Flow Chart –

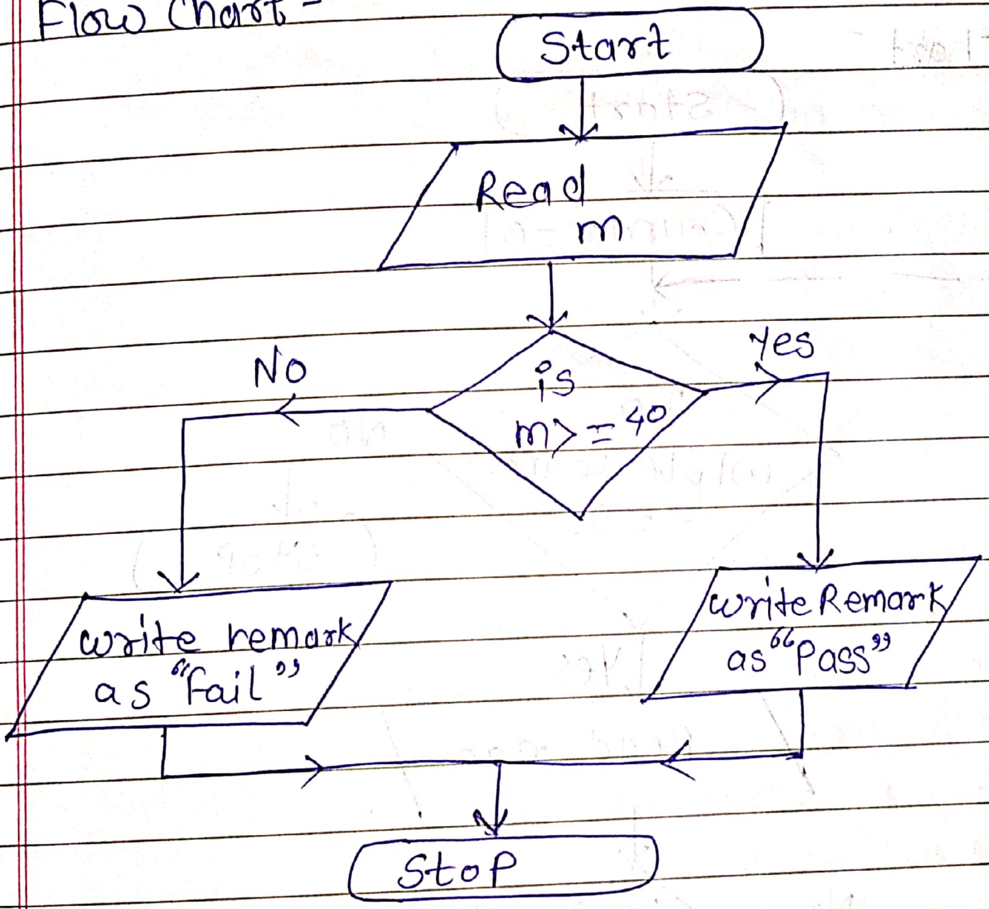


Q. Write an algorithm & draw a Flow Chart to Find the Student is Pass or Fail depend on ~~it's~~ marks obtain by Student

If marks $\geq$ 40 than Students is Passed otherwise Fail.

→ 1) Start the ~~programme~~ . Procedure.
2) Read marks of Student as m.
3) IF m >= 40 Then go to step 6.
4) Write Remark is "Fail".
5) go to step 7
6) write Remark is "Pass"
7) Stop.

Flow Chart -



Q.3. Write an algorithm & draw a flow Chart to Check the person is eligible or not eligible for voting. The Procedure is followed for 50 persons in the List.

→ 1) Start

2) Initially Counter = 0 ③ Do while Counter <=50

4) Read age.

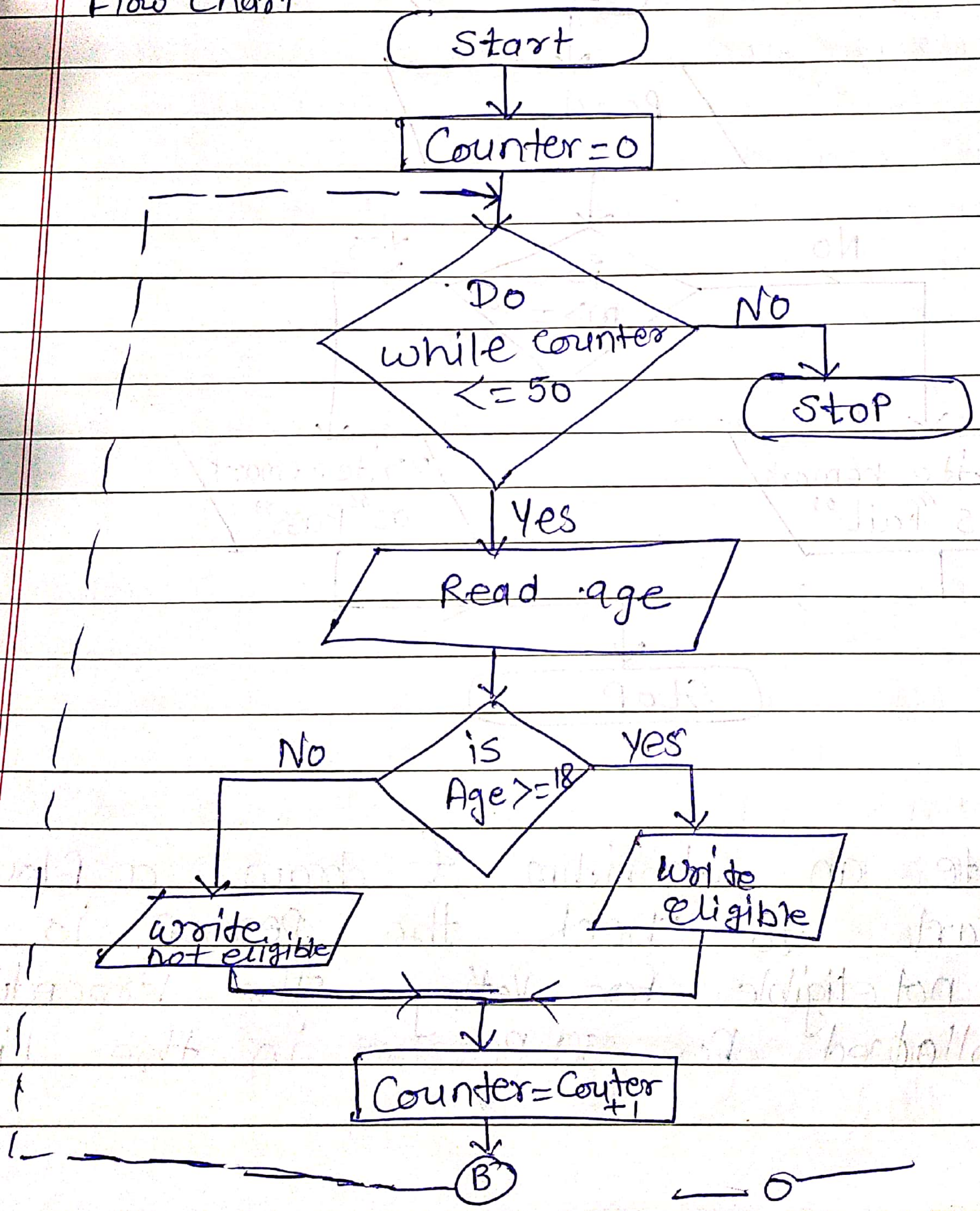
5) If age >=18 then Eligible else Not Eligible

6) Counter = Counter +1

7) go to step 3

8) Stop.

Flow Chart -


Start
Counter=0
Do while counter <=50
No
Stop
Yes
Read age
is Age>=18
No
yes
write not eligible
write eligible
Counter=Couter +1
B