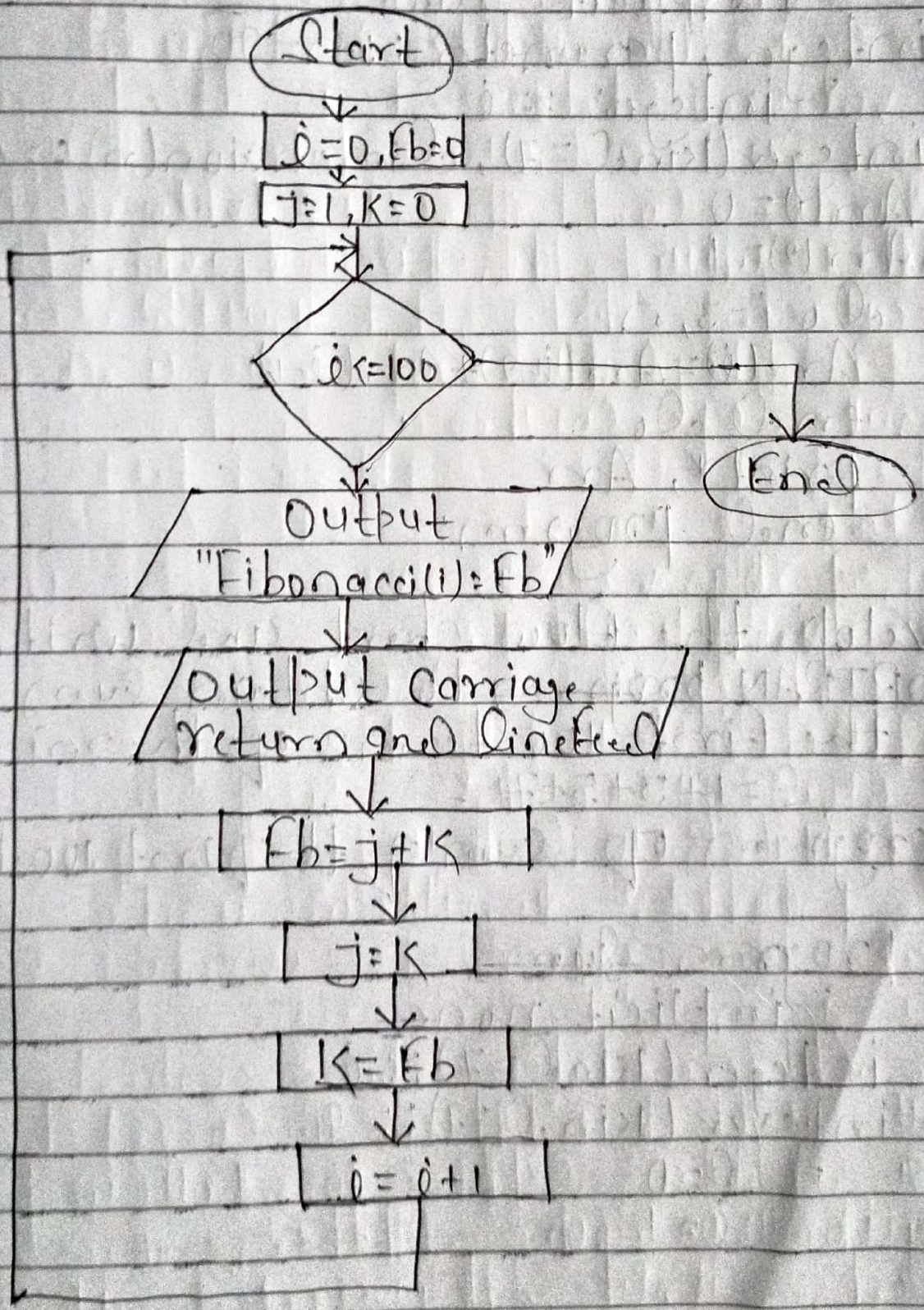
* Developed the flowchart and write FORTRAN program for generating Fibonacci series upto 100 terms.

Ans.



* Program of Fibonacci series.

```
program Fibonacci
  implicit none
integer, parameter :: n=90
    integer :: i
integer(kind=7), dimension(n) :: Arr
 Arr(1)=0
 Arr(2)=1
   do i=3, n
    Arr(i)=Arr(i-2)+Arr(i-1)
    end do
    print*, Arr
     end program
```

* Develop the flow chart and write a FORTRAN program to obtain sum of the first 100 terms of the series

$S = 1+3+5+7+\ldots$

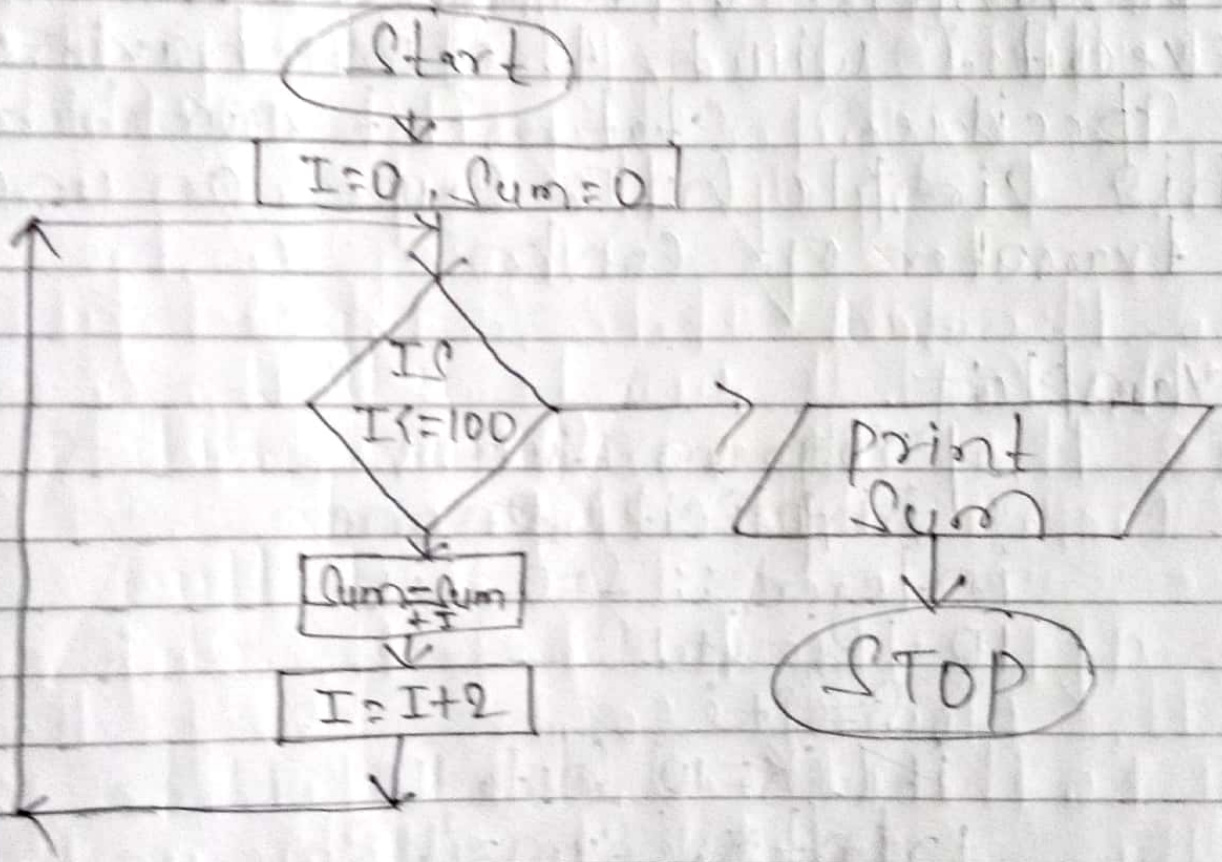
Ans.

Program of sum of the first 100 terms

```
program Sum
    implicit none
integer(kind=7):: i
integer(kind=7):: S
    S=0
    i=1
```

```
do while (i<=100)
   s= s+i
   i= i+2
end do
print*, 'sum of 100 terms=', s
end program sum
```

* Flowchart *



* With the help of suitable examples explain GOTO statement and IF statement in FORTRAN.

Ans. GOTO statement:-

⇒ The GOTO statement can be used to transfer control to another statement. The simplest form

Of the GoTo statement in Fortran is the unconditional branch. The statement takes the form:

GoTo n.

where n is the statement number of another statement in the program. When such a GoTo statement is encountered, the next statement executed will be the one having the specified statement number n. This simple GoTo causes an unconditional transfer of control.

Example:-

```
program sum
implicit none
integer :: s=0, i=0
10 i=i+1
   s=s+i
   if(i<=10) goto 10
print*, 'sum=', s
end program sum.
```

* If.... then statement:-

⇒ An if..... then statement consists of a logical expression followed by one or more statements and terminated by an end if

Statement. If the logical expression evaluates to true, then the block of code inside the if...then statement will be executed. If logical expression evaluates to false, then the first set of code after the end if statement will be executed.

Example:-

```
program ifprocheck
implicit none
integer :: n=10
    if(n<20) then
print*, "a if less than 20"
  end if
print*, "value of a is", a
  end program ifprocheck.
```

* Choosing suitable example explain the use of DO loop in FORTRAN.

Ans DO loop:-

⇒ The do loop construct enables a statement, or a series of statements to be carried out iteratively, while a given condition is true.

⇒ The flow of control for the do loop construct is as follows:-

1.) The initial step is executed first, and only once. This step allows us to declare and initialize any loop control variables. In our case, the variable var is initialized with the value start.

2.) Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement just after the loop. In our case, the condition is that the variable var reaches its final value stop.

3.) After the body of the loop executes, the flow of control jumps back up to the increment statement. This statement allows us to update the loop control variable var.

4.) The condition is now evaluated again. If it is true, the loop executes and the process repeats itself. After the condition becomes false, the loop terminates.

*Example:

```
program printNum
    implicit none
    integer :: N
    do N=11, 20
    print *, N
    end do
end program printNum
```

* Discuss in brief the role of compiler in computer.

Ans:

⇒ A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses. Typically, a programmer writes language statements in a language such as pascal or c one line at a time using an editor. The programmer then runs the appropriate language compiler, specifying the name of the file that contains the source statements.

⇒ When executing (running), the compiler first parses (or analyzes)

all of the language statements syntactically one after the other and then, in one or more successive stages or "passes", builds the output code, making sure that statements that refer to other statements are referred to correctly in the final code. Traditionally, the output of the compilation has been called object code or sometimes an object module.

⇒ A compiler is a computer program which helps we transform source code written in a high-level language into low-level machine language. It translates the code written in one programming language to some other language without changing the meaning of the code. The compiler also makes the end code efficient which is optimized for execution time and memory space.

⇒ The compiling process includes basic translation mechanisms and error detection.