CS programming Assign — 4

Program

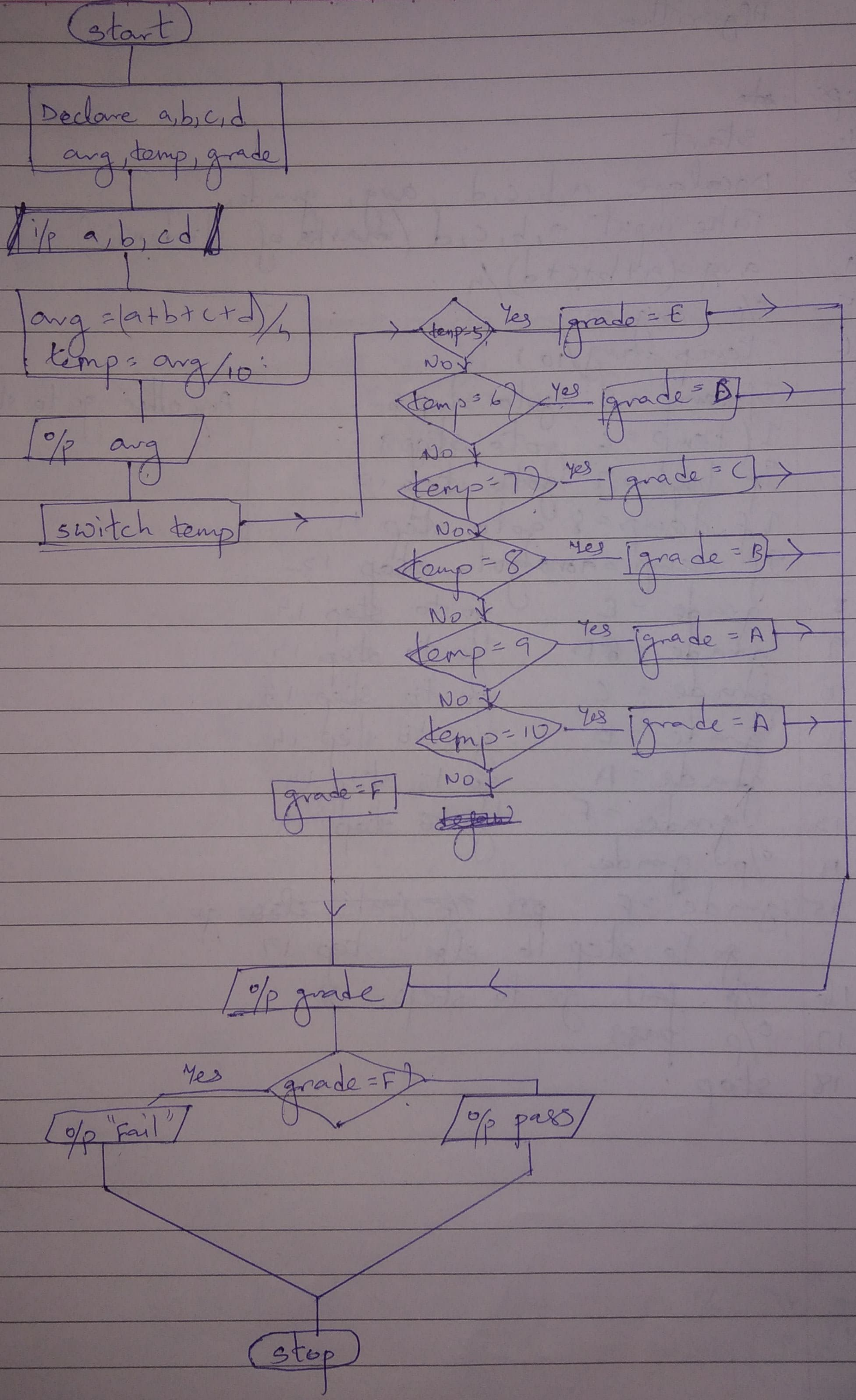
```
1) #include <stdio.h>
int main() {

int a,b,c,d;
float avg;
 char grade;
printf("Enter the ~~grade~~ marks out of 100 of four
       subjects ~~of ... #~~ \n");
scanf("%d %d %d %d", &a &b, &c ,&d);
avg = (float)(a+b+c+d)/4;
printf(" The average is %.1f \n", avg);
int temp = avg/10;

switch (temp) {

   case 5: grade = 'E';
           break;
   case 6: grade = 'D';
           break;
   case 7: grade = 'C';
           break;
   case 8: grade = 'B';
           break;
   case 9: ~~gra~~
   case 10: grade = 'A';
   break;
   default: grade = 'F';
  }
}

printf(" Final grade of the student is %c \n", grade);
grade == 'F' ? printf("Fail") : printf("Pass");
return0;
}
```

Algorithm.

steps : ~~st~~

```
1.  Start
2   Declare a,b,c,d , avg, grade.
3   Take input a,b,c,d (marks of students)
4   avg =(a+b+c+d)/4.
5.  o/p avg
6   temp = avg/10 ;
7   If temp = 5 goto step 8          | Any other go to step 13
    If temp = 6 goto step 9          |
    If temp = 7 goto step 10         |
    If temp = 8 goto step 11         |
    If temp = 9 or 10 goto step 12   |
8   grade = E        go to step 14
9   grade = D        go to step 14
10  grade = C        go to step 14
11  grade = B        go to step 14
12  grade = A        go to step 14
13  grade = F.       go to step 14
14  o/p grade.
15  If grade = F   ~~p~~ ~~o/p fail else~~ go
        go to step 16 else step 17
16  o/p fail  go to step 18
17  o/p pass
18  stop.
```

flowchart


start
Declare a,b,c,d
avg, temp, grade
i/p a, b, cd
avg =(a+b+c+d)/4
temp= avg/10
o/p avg
switch temp
temp=5
Yes
grade = E
NO
temp=6
Yes
grade= B
NO
temp=7
yes
grade = C
NO
temp=8
Yes
grade = B
NO
temp=9
Yes
grade = A
NO
temp=10
Yes
grade = A
NO
grade=F
~~default~~
o/p grade
Yes
grade=F
o/p "Fail"
o/p pass
stop

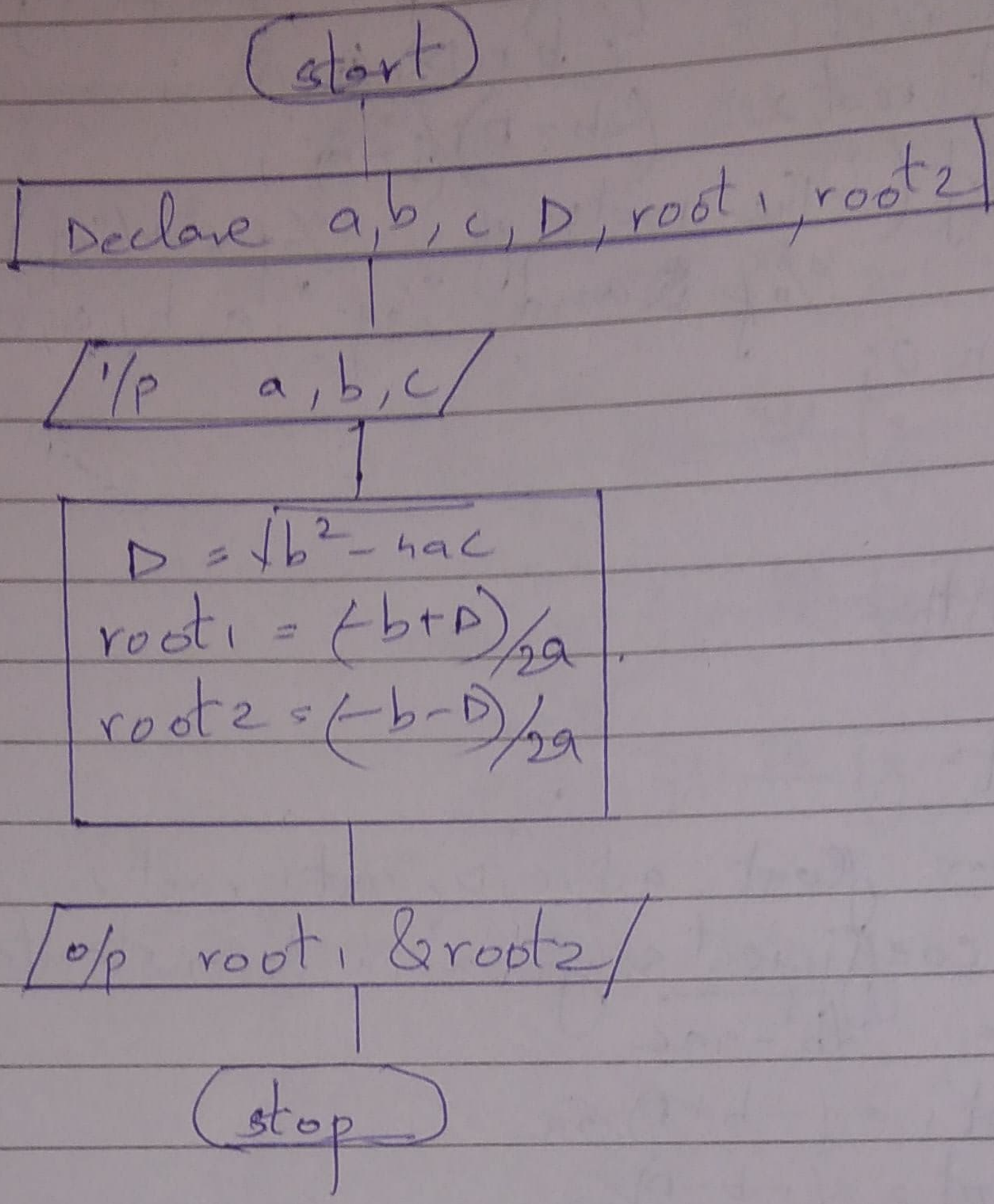
$x^2 + 4x + 4 = 0$

2)
```
#include <stdio.h>
#include <math.h>
int main(){
printf("Enter the coefficients of x^2 followed by x &
        constant \n");
float a,b,c;
scanf("%f %f %f",&a,&b,&c);
float D = sqrt(b*b - 4*a*c);
float root1 = (-b + D)/(2*a);
float root2 = (-b-D)/(2*a);
printf("The roots of %f x^2 + (%f)x + (%f) = 0 are
        %f ~~&~~ and %f",a,b,c,root1,root2);
return 0;
}
```

Algorithm -

```
Steps
1  start
2  Declare float a,b,c,D, root1, root2
3  i/p coefficients of $x^2$, x & constants as a,b,c
4    D = $\sqrt{b^2 - 4ac}$
5    root1 = $(-b+D)/2a$
6    root2 = $(-b-D)/2a$
7  o/p roots of $ax^2 + bx + c = 0$ are root1 & root2
8  stop.
```

flowchart


start
Declare a, b, c, D, root1, root2
I/p a, b, c
$D = \sqrt{b^2 - 4ac}$
$root1 = (-b+D)/2a$
$root2 = (-b-D)/2a$
O/p root1 & root2
stop

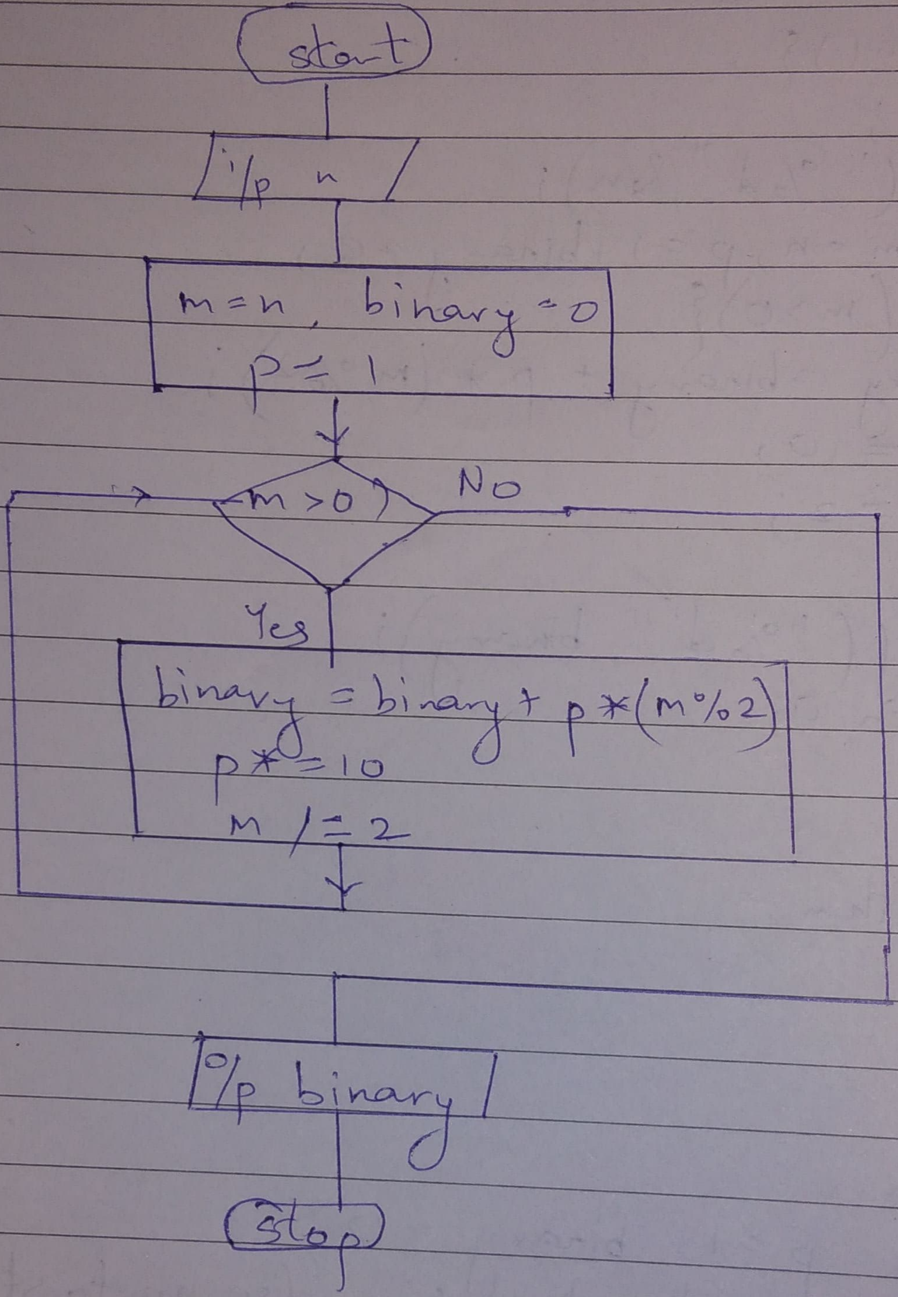
```
3) #include <stdio.h>
   int main(){
    int n;
    scanf(" %d ",&n);
    int m=n, p=1, binary=0;
   while (m>0){
     binary =binary + p* (m%2);
     p*=10;
     m/=2;
   }
    printf("%d", binary);
    return 0;
   }
```

Algorithm:

Steps:

```
1  start
2  i/p n.
3    m=n  p=1  binary=0
4  If m>0 go to step 5  else go to step 6
5  binary = binary + p* (m%2)
     p*=10
     m/=2
   go to step 4.
6  o/p binary
7  stop
```

flowchart

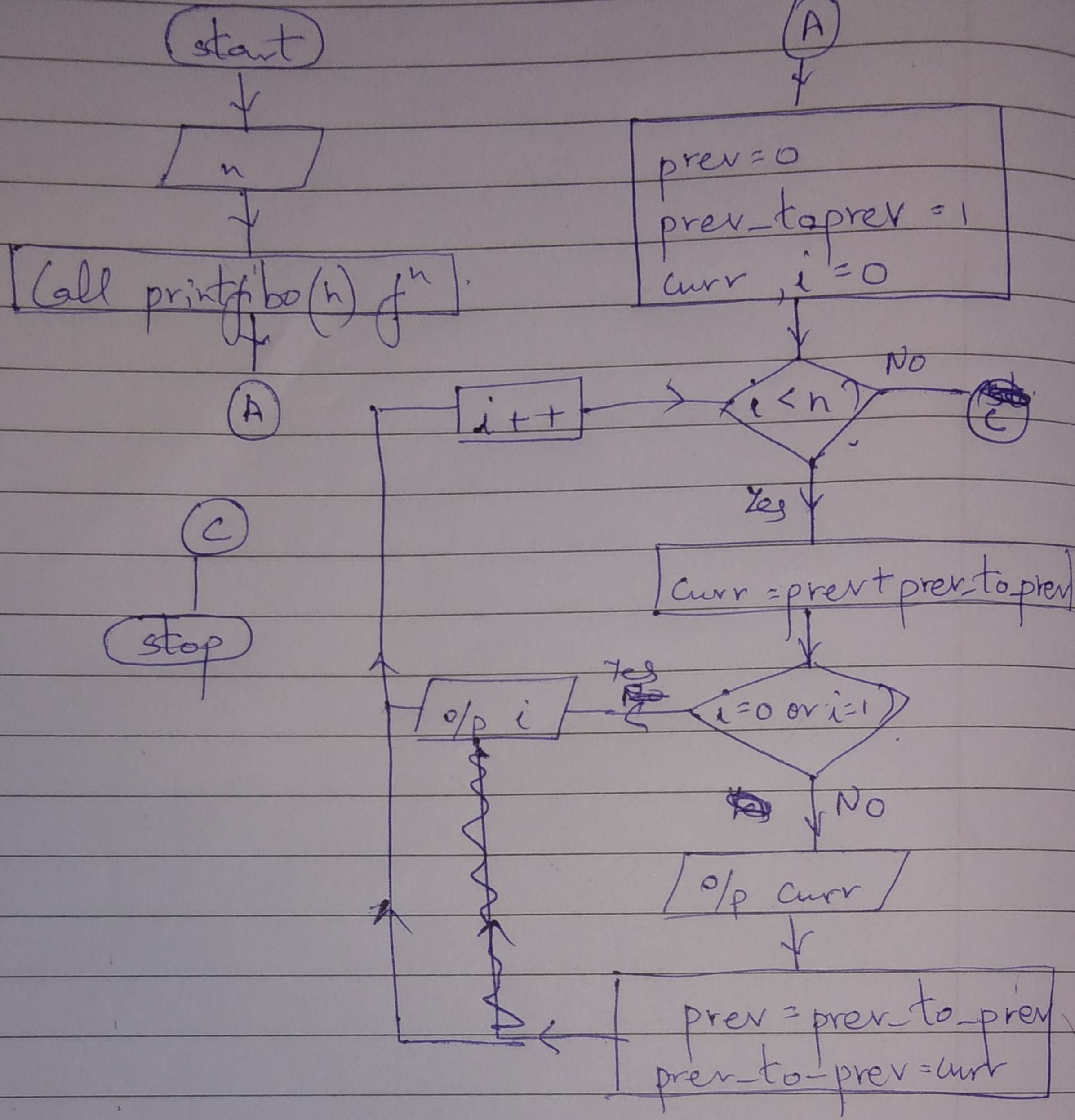

start
i/p n
m=n, binary=0
p=1
m>0?
No
Yes
binary = binary + p*(m%2)
p*=10
m/=2
o/p binary
stop

4)
```
#include <stdio.h>
void printfibo (int n){
  int prev = 0;
  int prev_to_prev = 1;
  int curr;
  for(int i = 0; i < n; i++){

     curr = prev + prev_to_prev;
     if (i == 0 || i == 1) printf(" %d ", i);
     else {
     printf("%d ", curr);
     prev = prev_to_prev;
     prev_to_prev = curr;
     }
   }
}

int main (){

   int n;
   scanf(" %d ", &n);
   printfibo (n);
   return 0;
}
```

Algorithm :

```
steps
1  Start
2  i/p n
3  Call printfibo(n) function.
4  return 0
5  stop;

   Algorithm of printfibo(n) function.
1  start with n as parameter.
2  Declare prev=0, prev_to_prev=1, curr, i=0
3  If i<n go to step 4 else go to step 8
4    curr = prev + prev_toprev;
5  If i=0 or i=1 ~~prin~~ o/p i ~~else~~ go to step 7
                     else go to step 6
6  print curr
   ~~tox~~ prev = prev_to_prev
       prev_to_prev= curr
7  i++ go to step 3

8  stop.
```

flowchart


start
n
Call printfibo(n) f^n
A
C
stop
A
prev=0
prev_toprev =1
curr , i =0
i<n
No
C
Yes
i++
curr =prev+prev_to_prev
i=0 or i=1
Yes
o/p i
No
o/p curr
prev = prev_to_prev
prev_to_prev =curr