Date No.

B) Looping Statements

Looping statements are the statements that are used to execute a block of statements for repeatedly until a particular condition is satisfied.

In C-programming, these are three looping statements:

(i) for loop
(ii) while loop
(iii) do-while loop.

} - This loops also contain nested structures.

Loop also means iteration and it is used to repeat some set of statement.

A loop has three part. They are as follows:
(i): Initialization
(ii): Termination condition.
(iii): Update expression; taking loop to next step.

Based on the placement of termination condition, loops are of two types:

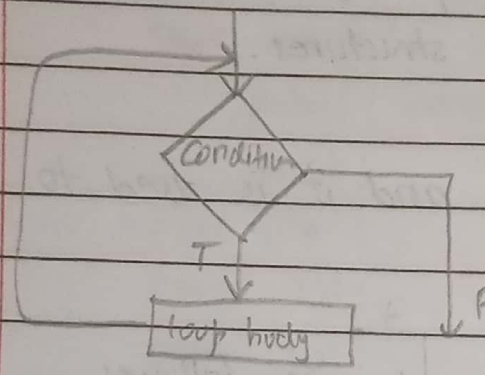
i) Entry-controlled loop: (ii) Exit-controlled loop:

i) Entry-controlled loop:
→ In Entry-controlled loop, termination statement is put when loop is about to start.
→ First, the condition is checked and then control enters ~~for~~ or exits the loop.

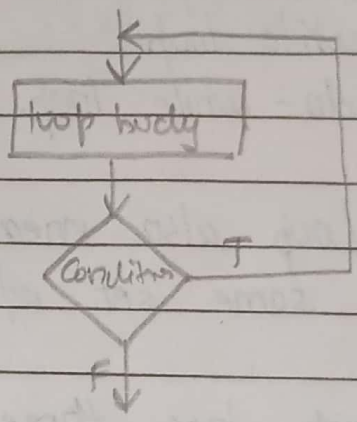
Date. No.

(ii) Exit-controlled loop:

- In exit-controlled loop, termination condition is put when the loop is about the end.
- First, the loop occurs, then the condition is checked which dictates entry or exit of the loop.



Entry-controlled loop.



Exit-controlled loop.

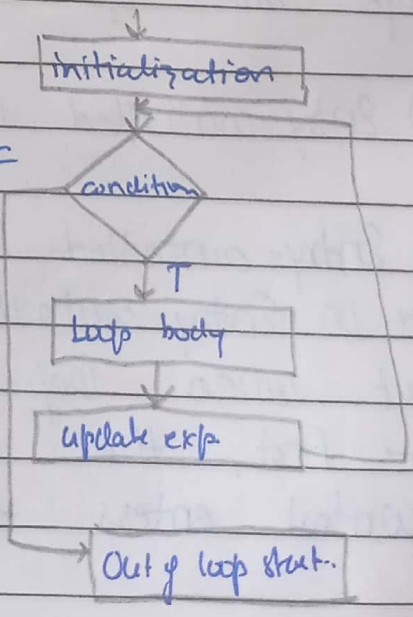
(i) For-loop

→ It is a entry controlled loop.

(x) Syntax:

```
for (exp 1; exp 2; exp 3)
{
    loop body statements;
}
out of loop statements;
```

(x) Flowchart



Date No.

(*) Working:

(i) Expression (1) initialization is done.
(ii) Expression (2) has condition which is checked and the control enters into loop. and is completed
(iii) Expression (3) ie, update expression is underwent
(iv) Again, control goes to exp (2) and condition is checked.
If true, loop continues in same manner
else, loop is terminated.

Note: i) Initialization is done only once.

Eg:
```
for (i=10; i<10, i++)
{ printf ("%d", i); }
printf ("Jk");
```
Output:
Jk
{ie, loop is not executed due to given termination condition}

(*) Properties:

(i): Only two semicolons are allowed and they are necessary.
Eg:
```
for (i=0; i<=1; i++)
    for ( ; ; )
```

(ii): Initialization is optional and can be done outside loop statement too.
Eg:
```
for ( ; i<=2, i++)
```
=> no output
```
i=1; for ( ; i<=2, i++)
```
=> 2 times loop continues.

Date. No.

(iii) : We can also do multiple initializations.

Eg: for (i=0, j=1; j<=1; j++);

↳ Here, double initization.

(iv): We can also write multiple conditions but condition writing is also optional.

Eg: for (i=0; ; i++) → infinite.

for (i=0, j<=0, i<=10; i++)

↳ Here, ~~second~~ last condition is considered looping conditions and conditions before them are just considered as statements.

(v): Multiple update systems or no update statements can also be done

eg: for (i=0; i<=1;); → infinite

for (i=0, i<=1, i++, j++); → Here, condition is checked for loop execution.

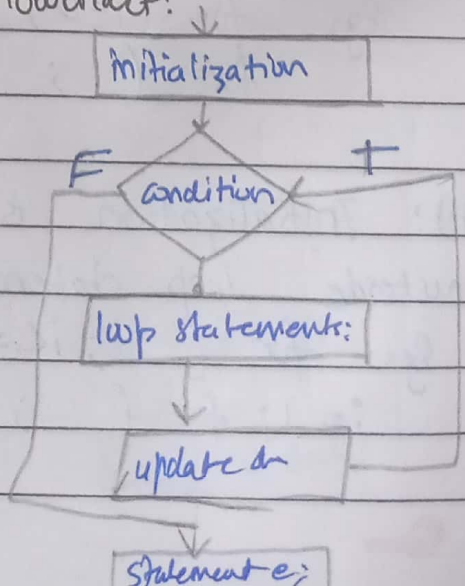
(ii) While - loop:

→ It is entry - controlled loop:

(*) Syntax:

```
while (condition)
{
    // loop body
    update exp
}
statement e;
```

(*) Flowchart:



Date. No.

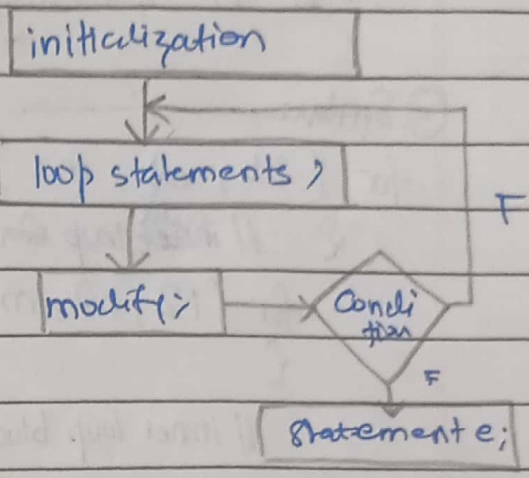
(iii): do-while loop:

→ It is exit-controlled loop.
→ This loop will run atleast once.

(*) Syntax:

```
do
{
  loop statements;
  modify;
}
while (condition);
statement e;
```

(*) Flowchart:



→ Use to run loop atleast once without checking condition.

(*) Differences between looping statements:

	for	while	do-while.
→	entry	entry	→ exit
→	for (ini-; condition; modify) { loop body }	while (condition) { loop body }	do { statements } while (condition);
→	used no of iterations are known	→ iteration no. not known	→ iteration no. not known.
→	control don't enter loop if false	→ control don't enter loop if false	→ Body of loop is executed atleast once

Date. No.

(*) Nesting of loops in C: Various loops can be nested.

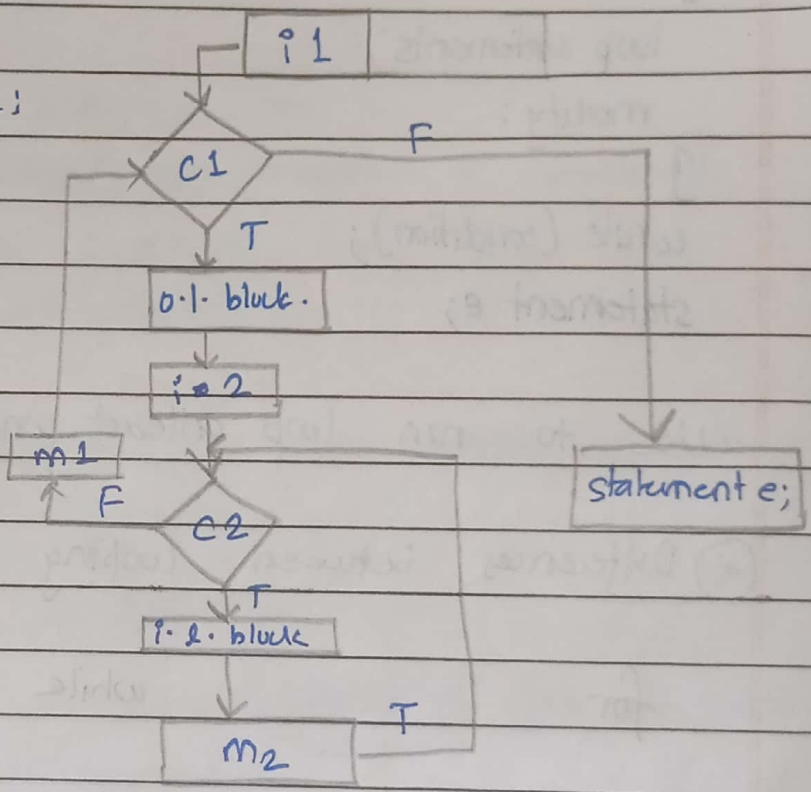
(i) Nested for-loop:

→ It is generally used for making patterns.

(*) Syntax:

```
for (i1, c1, m1)
{ // outer loop block;
  for (i2, c2, m2)
  {
    // inner loop block;
  }
}
statement e;
```

(*) Flowchart:

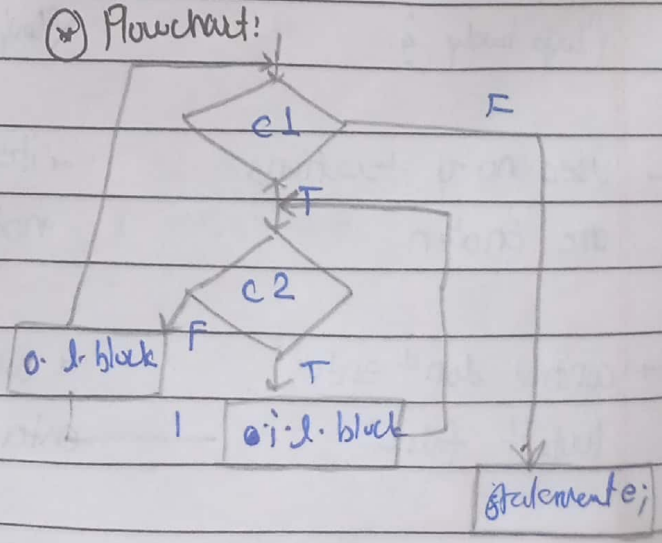


(ii) Nested while loop:

(*) Syntax:

```
while (c1)
{
  while (c2)
  { i.l. block }
  o.l. block
}
statement e;
```

(*) Flowchart:



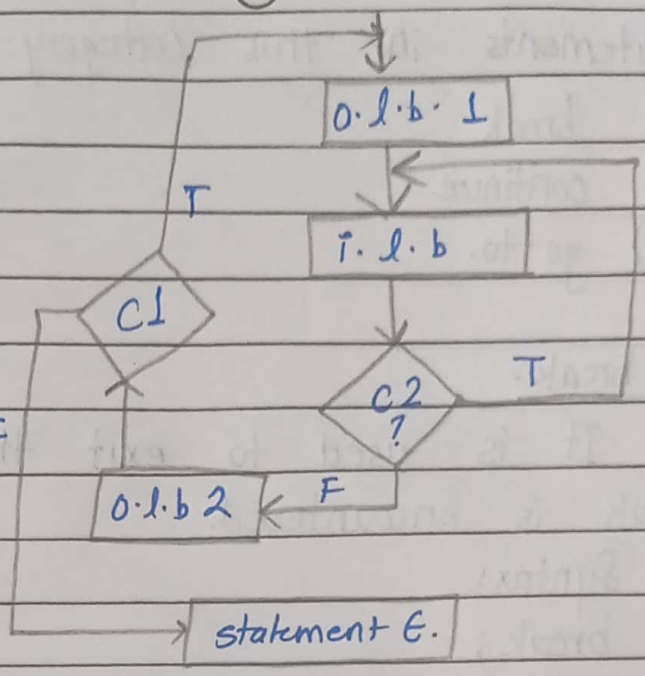
Date. No.

(iii) Nested do-while loop:

(*) Syntax:

```
do
{ //outer loop 1;
  do
  {
    i.l.block;
  }
  while (c2);
  o.l.block 2;
}
  while (c1);
statement e;
```

(*) Flowchart:



We can also do ~~looping in follow~~ nesting using different loops

Eg:

i)
```
while (c1)
{
  for (i2, c2, m2)
  {
    do
    { .... }
    while (c3);
  }
}
```

(ii): we can nest using ~~only~~ only two types of looping statements.