Decision - Making — Looping — Branching — Nesting

* Conditional Statements :

✓1. Simple if → 1 condition → T → execute
✓2. If - else → True { ✓ } → skip
False { ✓ } 2 conditions
✓3. If - else if - else   More than 2 conditions
4. Nested if - else
✓5. Ternary Operator (? :)
6. Switch Case

[Nesting or Branching]

(number)(int) check → even → is is div by 4

check → odd → is it div by 3

```
if {
  if {
  else {
}
else {
  if {
  else {
}
```

char → a - to - z or A to Z

Vowels : a, e, i, o, u & A, E, I, O, U

Consonants : (b - z) are (B - Z) consonants

Anything Else : (*, ., ?) → (invalid)

* Looping Statements :→ (Repetitions) (5 times - 10 times)

① while Loop
② do while loop
③ for loop
④ Nested loops

```
while ( condition) {      True ⇓
    statement;
    (inc / dec)
}
```

infinite loop

(0000)

```
do {
    enterPIN(); → (1, 2, 3) → done (1234)
}
while (pinIsWrong); → prompt  24 hrs
```

```
do { washClothes();
}
while (ClothesAre Dirty());
```

For Loop :→ Repeat for 10 times, 100 times, 200 times } no of repetitions known
↳ (finite)

```
for (initialization; condition; inc/dec) {
    statements;
}
```

```
#include<stdio.h>
int main(){
    for(int i=0; i<10; i++){
        printf("%d ", i);
    }
    return 0;
}
```

* Jump Statements :→

① break → terminates the loop
② continue → skips the current value & goes to the next value

Nested Loop : [Print a multiplication table 1 to 10]

for [i → variable] (iteration)

1×1 = 1   1×2 = 2 . . . .   1×10 = 10

(j variable)

i × j

1×1
1×2
1×3
⋮
1×10
2×1

} j will complete

i=1 j 1 to 10
i=2 "
i=3 "

10×1 = 10   10×10 = 100

```
for {
    for {
```

Identity Matrix   4×4 Matrix   (2D Array)

r == c
i == j

rows → / cols ↓	0	1	2	3
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

(HW - 1) n = 5

*				
*	*			
*	*	*		
*	*	*	*	
*	*	*	*	*

(HW - 2) n = 4

1			
1	2		
1	2	3	
1	2	3	4