C → 2 days   ESC (Ecma Script) 6+6 = 12
C++ → 2 days   6+6 = 12   } 24 hours   (DSA)   NBA AICTE NAAC

ANSI Standards

Founded : (1972) ← 32 years

Denis Ritchie : 1940 → A → +v1 → B +v2 → C

(1st Std Version) ANSI – C (1989)

UNIX

(Reva) ≈ VTU
(B.Tech)
B.E

CS/IT → TCS, IBM, Cognizant, Capgemini, DXC – FE

W, M — React — Openpyxl

P → bs4

{ C + Java — DSA
C + C++ + J + P — tuf, Striver
C + P, J, C++
J → selenium }

Data Types in C : → (Website – App) → Bugs

Primary	Void	Enumeration	Derived
Integer: short int 2 bytes (−128 to 127), 4 bytes (-2^{31} to $2^{31}-1$)	void	"enum" — Named Constants	(Arrays) → list [1, 2.0, "Reva"] (mutable)
Floating: float 4b, double 8b	INT-MAX / INT-MIN (long)	Jan – Dec, Mon – Sun	Strings, Pointers
Character: char → 1 byte		default – int, size – 4 bytes (H, L, M)	Struct, Union (Class)

* No built-in boolean datatype available in C.
* To utilize them we use #include <stdbool.h>

_name ✓   1name ✗ $name ✓   } Rules   } Naming Conventions: camelCase, Date Of Interest (vars, fun, methods)

Pascal Naming Convention — classes / constant

Operators : → operands ← a + b = c → operator → result; operator, expression

* Arithmetic Operators : +, −, /, *, %   Python: **, //
* Assignment Operators : =, +=, −=, /=, *=, %=
  ↳ Augmented Operators / Shorthand Operators
  [a = a+10] a += 10
* Logical Operators : &&, ||, !   and, or, not
* Comparison / Relational : >, <, >=, <=, ==, != (Boolean)
* Unary operators (++, --) Prefix / Postfix
* Ternary Operator (condition) ? tv : fv ; (shorthand if/else operator)
* (Bit Manipulation) (Bit Masking)
  Bitwise Operators

Bitwise: AND &, OR |, XOR ^, Right Shift >>, Left Shift <<, NOT ~

6 → 0110
8 → 1000
6|8 → 1110 (14)

int 5 → 0101
7 → 0111
5&7 → 0101 = 5 → int
int = int = int

8 → 1000
9 → 1001
8^9 → 0001 = 1

✓ S – 0
✓ D – 1

Bitwise Shift Operations : → (Value / step or unit)

a = 10
step = 2
(10 >> 2 = 2) ↓

| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | (added / discarded)
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | added

Leetcode   10 << 2 = 40

TV ~5 = 10 (2's c)
5 → 0101
~5 → 1010 = 10
5 → −6
[−6 = 10]
MCQ → ~5 : 1, 2, 10, (−6)

[0 – 1, 1 – 0]   [T – F, F – T]

abs(−6) = 6 → 0110
2's com { 1's c → 1001, +1 → 0001 } → 1010

~n Formula ~n = −n−1
(−500) = −(−500) − 1 = 499
500 = −501
TV – 2's com
PV – −n−1

n = 8, 4, 5, 9
[Power of 2]
if (n&(n−1) == 0) { }

8 → 1000, 0111, 8&7 0000
10000, 01111
4 → 0100, 0011, 0000 (4&3)
5 → 0101, 0100
9 → 1001, 1000

(n%2 == 0)
2 | 4 → 2, 0
Even or odd
if (n&1) == 1 odd, else even

12 → 1100, 0001 → 0
5 → 0101, 0001 → 1
8 → 1000, 0001 → 0
9 → 1001, 0001 → 1

Conditional Statements : →

(i) Simple if → only 1 condition
(ii) if - else → 2 conditions
(iii) if - else if - else ladder → More than 2
(iv) Nested if - else → Condition inside condition, Branching or nesting
(v) Switch → Better version of (3)
(vi) Ternary Operator

* Check whether a given character is a vowel or consonant using switch case : →
#include <ctype.h> to check if the character is an alphabet or not
[isalpha(ch)]

Looping Statements : →

(i) while loop (uncertain)
(ii) do while loop
(iii) for loop (finite)
(iv) Nested loops
(v) for-each loop or enhanced for loop (C++ & Java)

do {
  statement → at least once
} while (a > 100);   a = 10;   Exit Controlled Loop

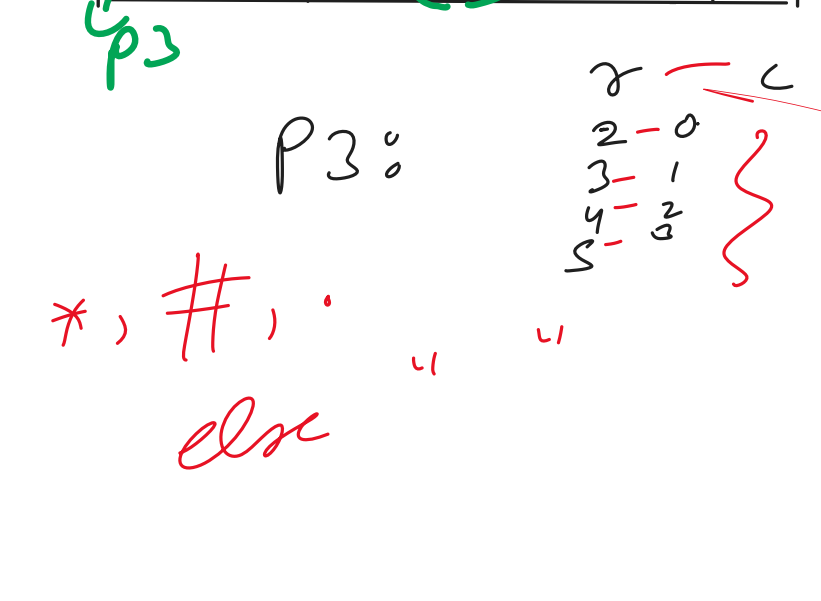
Entry Controlled → While ↳ false
Exit Controlled → Do While

while (a > 100) { statement; }   Entry controlled Loop

while (condition) {
  statements;
  inc / dec → (infinite loop)
}

Pattern Problems : → (Static Patterns : →)

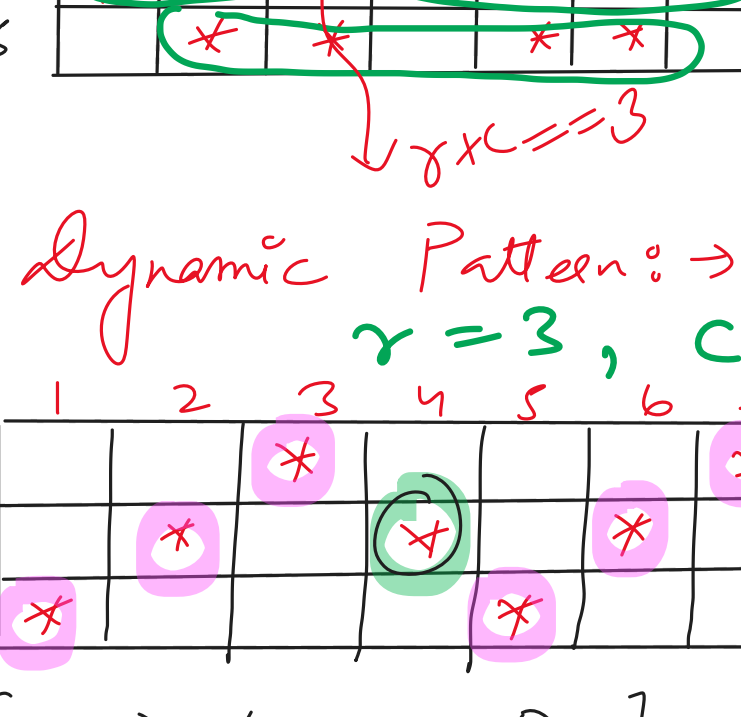
r = 6, c = 7



P1: r == 0
P2: r == 1
P3: r – c == 2 (2–0, 3–1, 4–2, 5–3) == 2
P4
*, #, ' ' else

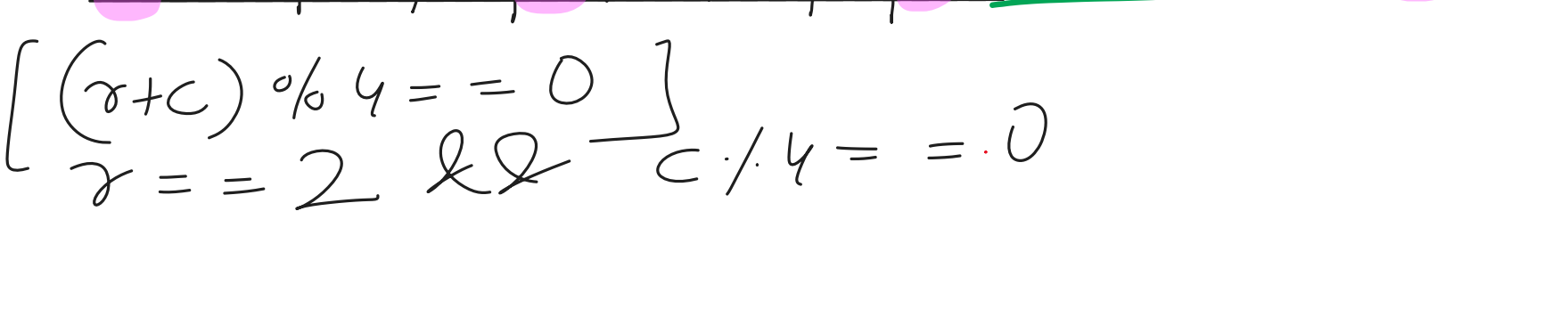
Rule : → Break the dia into similar parts
0 % 3 = 0
c 1, 2, 4, 5 → c % 3 != 0
c, 0, 3, 6 → c % 3 == 0
r c: 2+6, 3+5, 4+4 == 8

Reverse Heart Pattern : → Bosch (DS+ECE)



r == 5 && c % 3 != 0
r == 4 && c % 3 == 0
(r+c == 3)
{ r − c = −3 } or { c − r = 3 }
r+c == 3

Dynamic Pattern : → Zig Zag Pattern
r = 3, c = 9, 13, 17, 21, 25, . . .



[(r+c) % 4 == 0]
r == 2 && c % 4 == 0