04 June 2023 17:08

n=7

```
*  *  *     *  *  *
*  *           *  *
*                 *

*                 *
*  *           *  *
*  *  *     *  *  *
```

r	—	*
1	1	3
2	3	2
3	5	1
4	7	0
5	5	1

2, 3, 4

<= n/2 + 1

I
while(C){
U
}

for(I; C; U){
}

Data type

Primitive | Non - Primitive.

8 Types

← 4
- byte → 1 byte → 8 bit
- short 2 byte → 16 bit
- int 4 byte → 32 bit
- long 8 byte → 64 bit

$0 \leftrightarrow 2^8 - 1$

$2^8 = 256$

00 0000 00

$2^0 + 2^1 + \dots + 2^7$

$2^0 + 2^1 + 2^2 + \dots 2^7$

$\overline{9999..9}_n$

$(\_)_{10}$

90989
90990

90999
91000

Counting

0 1 2 3 4 5 6 7 8 9

10 11 12 13 : 19

99
100

$(\_)_5$

0 → 0.5

0–4

0 1 2 3 4 10 11 12 13 14 20

44
100

$(14)_5$

$4 \times 5^0 + 1 \times 5^1 = 9$

11111111

$2^0 + 2^1 + \dots 2^{n-1}$

~~$S(n) = 2^0 + 2^1 \dots 2^{n-1}$~~

$= 2^n - 1$

$2^3 \; 2^2 \; 2^1 \; 2^0$

$2^0 + 2^1 + . \; 2^3$

$2^0 + 2^1 + \dots 2^{n-1}$

$(\_)_3$ [0–2]

0	0	100	9
1	1	101	10
2	2	102	11
10	3	110	12
11	4	111	13
12	5	112	14
20	6	120	15
21	7	121	16
22	8	122	17
		200	

5681

$1 \times 10^0 +$
$8 \times 10^1 +$
$6 \times 10^2 +$
5×10^3

$(122)_3$

$2 \times 3^0 = 2 + 6 + 9$
$2 \times 3^1 +$
1×3^2

17

$(\ )_8 \rightarrow (\ )_{10}$

$\times 8^0$, $\times 8^1$, $\times 8^2$

(3) (6) 555

$5 \times 6^0 + 5 \times 6^1 + 5 \times 6^2$

Binary system

$(00)_2$	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

1011 → $2^0, 2^1, 2^2, 2^3$

$2^3 + 2^1 + 2^0$

$8 + 2 + 1$

$S(n) = 2^0 + 2^1 + 2^2 + \dots + 2^{n-1}$

$2 \cdot S(n) = 2^1 + 2^2 + 2^3 + \dots + 2^n$

$2S(n) - S(n) = S_n = 2^n - 2^0$

$2^n - 1$

$(14)_{10} \rightarrow (\ )_2$

2	14	0
2	7	1
2	3	1
2	1	1
	×	

1110

Divide

$(26)_{10} \rightarrow (\ )_2$

Scheduled

2	26	0
2	13	1
2	6	0
2	3	1
	1	

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$(1\ 1\ 0\ 1\ 0)$

$16\ 8\ 4\ 2\ 1$

$16 + 8 + 2 = 26$

$(43)_{10} \rightarrow (\ )_2$

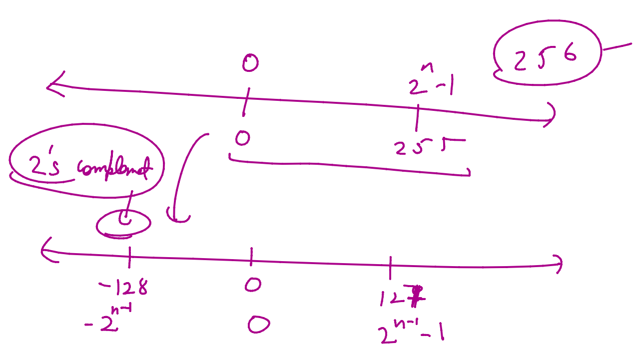
$(101011)_2$

$2^5\ \ 2^3\ 2^1\ 2^0$

$2^5 + 2^3 + 2^1 + 2^0$

$32 + 8$ $2 + 1$

256



0 to $2^{n}-1$.

-2 $\quad$ 0 $\quad$ $2^{n-1}-1$

0 to $2^n - 1$

pos → -2^{n-1} to $2^{n-1} - 1$

byte → 8 bit → -2^7 to $2^7 - 1$

-128 to 122

short (2 byte) → 16 bit → -2^{15} to $2^{15}-1$

$-32K$ to $32K$

$2^{15} \Rightarrow 2^{10} \cdot 2^5$ $\quad$ 1025 ≈ 1000

$\Rightarrow 1000 \cdot 2^5 \Rightarrow 32$

32K

int → 4 byte → 32 bit → -2^{31} to $2^{31}-1$

$2^{31} = 2 \cdot 2^{30} = 2 \cdot (2^{10})^3$

-2×10^9 to $2 \cdot 10^9$ $\quad \approx 2 \cdot (1000)^3$

$= 2 \cdot (10^3)^3$

$2 \cdot 10^9$

$2 \cdot 10^9$

$2 \times 10^1 = 20$

$2 \times 10^0 = 2$

$2 \times 10^9 = 10$

$2 \times 10^2 = 200 \rightarrow 3$

$2 \times 10^3 = 2000 \rightarrow 4$

---

log (8 byte) → 64 bit → -2^{63} to $2^{63}-1$

$2^{63} = 8 \times 2^{60} = (2^{10})^6 \times 8$

$= 10^{18} \times 8$

(32) bit

... 0 0 0 1 0 0

2^2 2^1 2^0

10^2 10^1 10^0

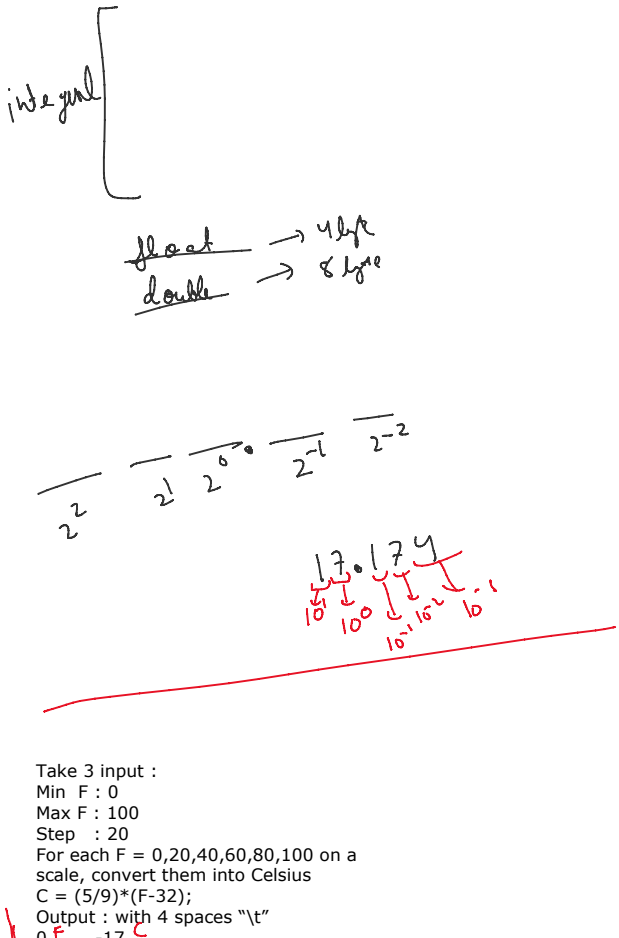
128

1 0 0 0 0 0 0 0

2^7 2^6

2's complement

-128



Take 3 input :
Min F : 0
Max F : 100
Step : 20
For each F = 0,20,40,60,80,100 on a scale, convert them into Celsius
C = (5/9)*(F-32);
Output : with 4 spaces "\t"

0 F -17 C
20 F -6 C
40 F 4 C
60 F 15 C
80 F 26 C
100 F 37 C