**DAY 10 - MISCELLANEOUS (BITWISE, MACROS, PREPROCESSORS) :**

// 1. Bitwise AND, OR, XOR

#include <stdio.h>

int main() {

int a = 5, b = 3;

printf("AND: %d\n", a & b);

printf("OR: %d\n", a | b);

printf("XOR: %d\n", a ^ b);

return 0;

}

// 2. Check even using bitwise

#include <stdio.h>

int main() {

int num;

scanf("%d", &num);

if(num & 1) printf("Odd");

else printf("Even");

return 0;

}

// 3. Toggle a bit

#include <stdio.h>

int main() {

int num, pos;

scanf("%d %d", &num, &pos);

num = num ^ (1 << pos);

printf("%d", num);

return 0;

}

// 4. Macro to find max of two numbers

#include <stdio.h>

#define MAX(a,b) ((a) > (b) ? (a) : (b))

int main() {

int x = 10, y = 20;

printf("%d", MAX(x, y));

return 0;

}

// 5. Macro for area of circle

#include <stdio.h>

#define AREA(r) (3.1415 \* (r) \* (r))

int main() {

float radius = 5.0;

printf("%.2f", AREA(radius));

return 0;

}

// 6. Preprocessor constant

#include <stdio.h>

#define PI 3.14159

int main() {

float r = 5.0;

printf("%.2f", PI \* r \* r);

return 0;

}

// 7. Calculator using switch

#include <stdio.h>

int main() {

char op;

float a, b;

scanf("%f %c %f", &a, &op, &b);

switch(op) {

case '+': printf("%.2f", a + b); break;

case '-': printf("%.2f", a - b); break;

case '\*': printf("%.2f", a \* b); break;

case '/': printf("%.2f", a / b); break;

default: printf("Invalid operator");

}

return 0;

}

// 8. Binary to decimal

#include <stdio.h>

int main() {

long bin;

int dec = 0, base = 1;

scanf("%ld", &bin);

while(bin > 0) {

dec += (bin % 10) \* base;

bin /= 10;

base \*= 2;

}

printf("%d", dec);

return 0;

}

// 9. Basic stack using array

#include <stdio.h>

#define SIZE 5

int stack[SIZE], top = -1;

void push(int val) {

if(top == SIZE-1) return;

stack[++top] = val;

}

int pop() {

if(top == -1) return -1;

return stack[top--];

}

int main() {

push(10); push(20); push(30);

printf("%d ", pop());

printf("%d ", pop());

return 0;

}

// 10. Basic queue using array

#include <stdio.h>

#define SIZE 5

int queue[SIZE], front = -1, rear = -1;

void enqueue(int val) {

if(rear == SIZE-1) return;

if(front == -1) front = 0;

queue[++rear] = val;

}

int dequeue() {

if(front == -1) return -1;

int val = queue[front];

if(front == rear) front = rear = -1;

else front++;

return val;

}

int main() {

enqueue(10); enqueue(20); enqueue(30);

printf("%d ", dequeue());

printf("%d ", dequeue());

return 0;

}