Worksheet - 1

▼ Level 1 : Basics & Control Structures (10 Problems)

1. Print all prime numbers between 1 and N.

```
#include <stdio.h>
#include <math.h>

void printPrime(int N);

int main() {

    int N;
    printf("Enter the value of N: ");
    scanf("%d", &N);

    printPrime(N);

    return 0;
}

void printPrime(int N) {

    for (int i = 2; i<=N; i++) {

        int isPrime = 1;

        for (int j = 2; j<=sqrt(i); j++) {
            if (i%j==0) {
                isPrime = 0;
            }
        }
        if(isPrime) {
            printf("%d\n", i);
        }
    }
}
```

2. Print Fibonacci Series up to N terms.

```
#include <stdio.h>

void printFibo(int N);

int main() {

    int N;
    printf("Enter the value of N: ");
    scanf("%d", &N);

    printFibo(N);

    return 0;
}

void printFibo(int N) {
    int n1 = 0, n2 = 1, c = 0, nth;
    if (N<0) {
        printf("Enter a positive value for N");
    } else if (N == 0) {
        printf("Fibonacci Sequence: \n");
        printf("%d", N);
    } else {
        printf("Fibonaci Sequence: \n");
        while (c<=N) {
            printf("%d ", n1);
            nth = n1+n2;
            n1 = n2;
            n2 = nth;
            c++;
        }
    }

}
```

3. Reverse a given number

```
#include <stdio.h>

void revNum(int N);

int main() {

    int N;
    printf("Enter the value for N: ");
    scanf("%d", &N);
    revNum(N);

    return 0;
}

void revNum(int N) {
    printf("Original Number: %d\n", N);
    int last_digit, rev;
    while (N>0) {
        last_digit = N%10;
        rev = rev * 10 + last_digit;
        N /= 10;
    }
    printf("Reversed Number: %d", rev);
}
```

4. Check if a number is Armstrong or Not

```
#include <stdio.h>
#include <math.h>

void checkArm(int N);

int main() {

    int N;
    printf("Enter the value for N: ");
    scanf("%d", &N);

    checkArm(N);

    return 0;
}

void checkArm(int N) {
    int last_digit, x = 0, c=0, temp;
    temp = N;
    while (temp>0) {
        temp /= 10;
        c++;
    }
    temp = N;
    while (N>0) {
        last_digit = N%10;
        x += pow(last_digit, c);
        N /= 10;
    }
    if (x == temp) {
        printf("The given number %d is an Armstrong Number", x);
    } else {
        printf("The given number %d is not an Armstrong Number", x);
    }
}
```

5. Find GCD and LCM of two numbers

```
#include <stdio.h>

int findGCD(int N1, int N2);
int findLCM(int N1, int N2, int gcd);

int main() {

    int N1, N2, gcd_val, lcm_val;
    printf("Enter the values of two numbers: ");
    scanf("%d %d", &N1, &N2);

    gcd_val = findGCD(N1, N2);
    lcm_val = findLCM(N1, N2, gcd_val);

    printf("GCD of %d and %d: %d\n", N1, N2, gcd_val);
    printf("LCM of %d and %d: %d\n", N1, N2, lcm_val);

    return 0;
}

int findGCD(int N1, int N2) {
    while (N2 != 0) {
        int temp = N2;
        N2 = N1 % N2;
        N1 = temp;
    }
    return N1;
}

int findLCM(int N1, int N2, int gcd){
    return (N1*N2)/gcd;
}
```

6. Convert decimal to binary without using array / string.

```
#include <stdio.h>

void deciTobin(int n);

int main() {

    int n;
    printf("Enter a number: ");
    scanf("%d", &n);

    deciTobin(n);

    return 0;
}

void deciTobin(int n) {
    if (n>1) {
        deciTobin(n/2);
    }
    printf("%d", n%2);
}
```

7. Print the following pattern:

   *

   **

```
#include <stdio.h>

void printPat();
```

```
int main() {
    printPat();
    return 0;
}

void printPat() {
    for (int i=1; i<=2; i++) {
        for (int j = 1; j<=i; j++) {
                    printf("*");
        }
        printf("\n");
    }
}
```

8. Count the number of the digits, sum of digits and reverse of a number in a single program

```
#include <stdio.h>

int no_of_digits(int N);
void sum_of_digits(int N);
void revNum(int N);

int main() {

    int n;
    printf("Enter the value of n: ");
    scanf("%d", &n);

    int count = no_of_digits(n);
    printf("No of digits of %d: %d\n", n, count);
    sum_of_digits(n);
    revNum(n);

    return 0;
}

int no_of_digits(int N) {
    int count = 0;

    while (N>0) {
        N/=10;
        count++;
    }

    return count;
}

void sum_of_digits(int N) {
    int last_digit, sum = 0;
    int temp = N;
    while(N>0) {
        last_digit = N % 10;
        sum += last_digit;
        N /= 10;
    }

    printf("Sum of digits of %d: %d\n", temp, sum);
}

void revNum(int N) {
    int last_digit, rev = 0;
    while (N>0) {
        last_digit = N % 10;
        rev = rev * 10 + last_digit;
        N /= 10;
    }

    printf("Reversed Number: %d", rev);
}
```

9. Check whether the given year is leap year or not.

```
#include <stdio.h>
#include <stdbool.h>

bool isLeap(int year);

int main() {

    int year;
    printf("Enter a year: ");
    scanf("%d", &year);

    if (isLeap(year)) {
        printf("The given year %d is a leap year", year);
    } else {
        printf("The given year %d is not a leap year", year);
    }

    return 0;
}

bool isLeap(int year) {
    if ((year%4 == 0 && year%100 != 0) || (year%400 == 0)) {
        return true;
    }  else {
        return false;
    }
}
```

10. Print multiplication table of a given number up to 10.

```
#include <stdio.h>

void print10x(int N);
```

```
int main() {

    int n;
    printf("Enter a number: ");
    scanf("%d", &n);

    print10x(n);

    return 0;
}

void print10x(int N) {
    int val;
    for (int i = 1; i <= 10; i++) {
        val = N * i;
        printf("%d x %d = %d\n", N, i, val);
    }
}
```

▼ Level 2 : Functions (10 Problems)

11. Write a function to check if a number is prime (return 1 if prime, 0 otherwise).

```
#include <stdio.h>
#include <math.h>

int isPrime(int N);

int main() {
    int n, flag;
    printf("Enter a number: ");
    scanf("%d", &n);

    flag = isPrime(n);
    printf("%d", flag);
}

int isPrime(int N) {

    if (N<=1) {
        return 0;
    }

    for (int i = 2; i<=sqrt(N); i++) {
        if (N%i==0) {
            return 0;
        }
    }

    return 1;
}
```

12. Write a function to calculate factorial using recursion.

```
#include <stdio.h>

int calcFac(int N);

int main() {

    int n;
    printf("Enter a number: ");
    scanf("%d", &n);

    printf("Factorial of %d: %d", n, calcFac(n));

    return 0;
}
```

```
int calcFac(int N) {

    if (N == 1 || N == 0) {
        return 1;
    }

    return N * calcFac(N-1);
}
```

13. Write a function to swap two numbers without using a third variable.

```
#include <stdio.h>

void swapNums(int a, int b);

int main() {

    int n1, n2;
    printf("Enter the values for n1 and n2: ");
    scanf("%d %d", &n1, &n2);

    printf("Before Swapping: %d and %d\n", n1, n2);
    swapNums(n1,n2);

    return 0;
}

void swapNums(int a, int b) {
    a = a + b;
    b = a - b;
    a = a - b;
    printf("Swapped Numbers: %d and %d", a, b);
}
```

14. Write a function to return the power of a number (x ^ y) without using pow()

```
#include <stdio.h>

double calcPow(int a, int b);

int main() {

    int x, y;
    printf("Enter the values for x and y: ");
    scanf("%d %d", &x, &y);

    printf("%g", calcPow(x,y));

    return 0;
}

double calcPow(int a, int b) {

    if (b == 0)
        return 1;

    double p = 1;
    int exp = b;  // keep original exponent

    if (b < 0)
        b = -b;

    while (b--) {
        p *= a;
    }

    if (exp < 0)
        return 1.0 / p;
    return p;
}
```

15. Write a menu-driven program using functions for: a) Factorial b) Prime check c)
    Odd/Even d) Exit

```
#include <stdio.h>

int calcFac(int n);
int isPrime(int n);

int menu();

int main() {
    menu();
    return 0;
}

int menu() {
    char opt;
    int N;

    while (1) {

        printf("\n--- MENU ---\n");
        printf("A) Factorial\n");
        printf("B) Check Prime\n");
        printf("C) Odd / Even\n");
        printf("D) Exit\n");
        printf("Enter your choice: ");
        scanf(" %c", &opt);   // space before %c

        switch (opt) {
```

```
case 'A':
case 'a':
    printf("Enter a number: ");
    scanf("%d", &N);
    printf("Factorial = %d\n", calcFac(N));
    break;

case 'B':
case 'b':
    printf("Enter a number: ");
    scanf("%d", &N);

    if (isPrime(N))
        printf("It is Prime\n");
    else
        printf("Not Prime\n");
    break;

case 'C':
case 'c':
    printf("Enter a number: ");
    scanf("%d", &N);

    if (N % 2 == 0)
        printf("Even Number\n");
    else
        printf("Odd Number\n");
    break;

case 'D':
case 'd':
    printf("Exiting Program...\n");
    return 0;

default:
    printf("Invalid choice! Try again.\n");
```

```
        }
    }
}

// Recursive factorial function
int calcFac(int n) {
    if (n == 0 || n == 1)
        return 1;
    return n * calcFac(n - 1);
}

// Prime checking
int isPrime(int n) {
    if (n <= 1)
        return 0;

    for (int i = 2; i * i <= n; i++)
        if (n % i == 0)
            return 0;

    return 1;
}
```

16. Write a function to check if a number is palindrome.

```
#include <stdio.h>
#include <stdbool.h>

bool checkPal(int N);

int main() {

    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    if (checkPal(n)) {
        printf("The given number %d is a palindrome", n);
    } else {
        printf("The given number %d is not a palindrome", n);
    }

    return 0;
}

bool checkPal(int N) {
    int temp = N;
    int last_digit, rev;
    while (N>0) {
        last_digit = N % 10;
        rev = rev * 10 + last_digit;
        N /= 10;
    }
    if (temp == rev) {
        return true;
    } else {
        return false;
    }
}
```

17. Write a function to find the sum of first N natural numbers using recursion.

```
#include <stdio.h>

int sumNat(int N);

int main() {

    int n;
    printf("Enter a number: ");
    scanf("%d", &n);

    int sum = sumNat(n);

    printf("Sum of first %d natural numbers: %d", n, sum);

    return 0;
}

int sumNat(int N) {
    if (N>=1) {
        return N+sumNat(N-1);
    } else {
        return 0;
    }
}
```

18. Write a function to convert temperature from Celsius to Fahrenheit and vice versa.

```
#include <stdio.h>

int main() {

    int choice;
    double f, c, t;

    printf("1) To convert Celsius to Fahrenheit\n");
    printf("2) To convert Fahrenheit to Celsius\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch(choice) {
        case 1: // Celsius to Fahrenheit
            printf("Enter a temperature in Celsius: ");
            scanf("%lf", &t);

            f = t * (9.0/5.0) + 32;
            printf("The temperature in Fahrenheit: %g\n", f);
            break;

        case 2: // Fahrenheit to Celsius
            printf("Enter a temperature in Fahrenheit: ");
            scanf("%lf", &t);

            c = (t - 32) * (5.0/9.0);
            printf("The temperature in Celsius: %g\n", c);
            break;

        default:
            printf("Invalid choice!");
    }

    return 0;
}
```

19. Write a function that returns the reverse of a number (do not print inside function).

```
#include <stdio.h>

int revNum(int N);

int main() {

    int n;
    printf("Enter a number to be reversed: ");
    scanf("%d", &n);

    printf("Original Number: %d", n);
    printf("Reversed Number: %d", revNum(n));

    return 0;
}

int revNum(int N) {
    int rev = 0, last_digit = 0;
    while (N>0) {
        last_digit = N % 10;
        N /= 10;
        rev = rev * 10 + last_digit;
    }
    return rev;
}
```

20. Write a recursive function to print Fibonacci series.

```
#include <stdio.h>

int fibo(int N);

int main() {

    int n;
    printf("Enter a number: ");
    scanf("%d", &n);

    for (int i = 0; i<=n; i++) {
        printf("%d ", fibo(i));
    }

    return 0;
}

int fibo(int N) {
    if (N == 0) {
        return 0;
    } else if (N == 1) {
        return 1;
    } else {
        return fibo(N-1) + fibo(N-2);
    }
}
```

Basics & Logic Building

1. Library Book ID Generator The college library assigns a unique 6-digit book ID to every
new book. A librarian enters a number N (100000 ≤ N ≤ 999999). The system must
check whether N is a "Beautiful Book ID" — i.e., the sum of the first three digits equals
the sum of the last three digits. Print "Yes" if beautiful, "No" otherwise.
Constraints:
100000 ≤ N ≤ 999999

```
/* Takes a 6-digit Number → Adds the 1st 3 digits and Last 3 digits
Print Yes if both are equal otherwise No */

#include <stdio.h>

int checkBeau_Book(int N);

int lb_id_gen();

int main() {

    int book_id = lb_id_gen();
    checkBeau_Book(book_id);

    return 0;
}

int lb_id_gen() {
    int N;
    printf("Enter a Library Book ID: (6-digit number): ");
    scanf("%d", &N);
    return N;
}

int checkBeau_Book(int N) {

    int last_digit1 = 0;
    int last_digit2 = 0;
    int f3 = 0, l3 = 0, s1 = 0, s2 = 0;

    f3 = N/1000;
    l3 = N % 1000;

    for (int i = 0; i<3; i++) {
        last_digit1 = f3 % 10;
        last_digit2 = l3 % 10;
        f3 /= 10;
        l3 /= 10;
        s1 += last_digit1;
        s2 += last_digit2;
    }

    if (s1 == s2) {
        printf("Yes");
    } else {
        printf("No");
    }

}
```

2. Electricity Bill Calculator – Tier System A city follows a special electricity billing rule:
   0–100 units: ₹4/unit 101–200 units: ₹4.5/unit 201–400 units: ₹5/unit Above 400: ₹6/unit
   Given the number of units consumed by a customer, calculate and print the total bill
   amount (rounded to 2 decimal places). Constraints: 1 ≤ units ≤ 10000

```
/* 0-100 → ₹4; 101-200 → ₹4.5; 201-400 → ₹5; Above 400 → ₹6 */

#include <stdio.h>

double calcBill(int N);

int main() {

    int n;
    printf("Enter the units consumed by the consumer: ");
    scanf("%d", &n);

    printf("Total Bill for %d units: %.2lf", n, calcBill(n));

    return 0;
}

double calcBill(int N) {
    double total = 0.0;

    if (N > 400) {
        total += (N - 400) * 6;
        N = 400;
    } if (N > 200) {
        total += (N-200) * 5;
        N = 200;
    } if (N > 100) {
        total += (N - 100) * 4.5;
        N = 100;
    }
    total += N * 4;

    return total;
}
```

3. ATM PIN Validation Story A bank allows only 3 attempts to enter the correct 4-digit PIN.
   Write a program that reads the correct PIN first, then keeps reading user attempts. If the
   correct PIN is entered within 3 tries, print "Welcome!", else print "Card Blocked".
   Constraints: 1000 ≤ PIN ≤ 9999

```
#include <stdio.h>

// Function to validate PIN with 3 attempts
void validatePIN(int correctPIN) {
    int attemptPIN;
    int attempts = 3;

    while (attempts > 0) {
        printf("Enter PIN: ");
        scanf("%d", &attemptPIN);

        if (attemptPIN == correctPIN) {
            printf("Welcome!\n");
            return;  // Exit the function if correct
        }

        attempts--;
    }

    // If all attempts are used
    printf("Card Blocked\n");
}

int main() {
    int correctPIN;

    printf("Enter correct 4-digit PIN (1000–9999): ");
    scanf("%d", &correctPIN);

    validatePIN(correctPIN);

    return 0;
}
```

▼ Functions & Recursion

4. Medicine Expiry Checker A pharmacy receives medicines with manufacturing month
   and year (MM YYYY). They use a function isExpired(currentMonth, currentYear,
   mfgMonth, mfgYear) that returns 1 if the medicine is expired (shelf life = 24 months from
   manufacturing month), otherwise 0. Write this function and use it in main to check 5
   medicines. Constraints: 1 ≤ Month ≤ 12, 2000 ≤ Year ≤ 2030

```
// 24 months = 2 years from mfg

#include <stdio.h>

int isExpiry(int curMonth, int curYear, int mfgMonth, int mfgYear);

int main() {

    int month = 0, year = 0;
    int curMonth = 12, curYear = 2025;
    printf("Enter the Manufacturing Month: ");
    scanf("%d", &month);

    printf("Enter the Manufacturing Year: ");
    scanf("%d", &year);

    printf("%d",isExpiry(12, 2025, month, year));

    return 0;
}

int isExpiry(int curMonth, int curYear, int mfgMonth, int mfgYear) {
    if ((curYear - mfgYear) > 2) {
        return 1;
    } else if ((curYear - mfgYear) == 2) {
        if (curMonth > mfgMonth) {
            return 1;
        } else {
            return 0;
        }
    } else {
        return 0;
    }
}
```

5. Employee Bonus Calculator HR needs a reusable function calculateBonus(yearsOfService, salary) that returns the bonus amount: ≥10 years →
   50% of salary 6–9 years → 30% of salary <6 years → 10% of salary Write the function
   and print the final salary (salary + bonus) for N employees. Constraints: 1 ≤ N ≤ 100, 0 ≤
   yearsOfService ≤ 40, 30000 ≤ salary ≤ 200000

```
/* y >= 10 → 50%; y in 6-9 → 30%; less than 6 → 10%
   Display total salary for N employees*/

#include <stdio.h>

double calcBonus(int yearsOfService, double salary);

int main() {

    int N, years; // No. of Employees
    double sal;

    printf("Enter the no. of employees: ");
    scanf("%d", &N);

    for (int i = 0; i<N; i++) {
        printf("Enter the Salary of the employee: ");
        scanf("%lf", &sal);

        printf("Enter the years of service of the employee: ");
        scanf("%d", &years);

        double bonus = calcBonus(years, sal);
        double total = sal + bonus;

        printf("Final Salary: %.2lf\n", total);
    }
    return 0;
}

double calcBonus(int yearsOfService, double salary) {

    double bonus = 0.0;

    if (yearsOfService >= 10) {
        bonus = 0.50 * salary;
    } else if ((yearsOfService >= 6) && (yearsOfService <= 9)) {
        bonus = 0.30 * salary;
    } else if (yearsOfService < 6) {
        bonus = 0.10 * salary;
    }

    return bonus;
}
```

▼ 1D Arrays

6. Metro Train Seat Allocation A metro coach has 72 seats arranged in a row. Due to
   reservation, some seats are already occupied (represented by 1) and some are vacant
   (0). Find and print the maximum number of consecutive vacant seats available for a
   group. Example: 1 0 0 1 0 0 0 1 → Answer: 3 Constraints: 1 ≤ N ≤ 1000

```
#include <stdio.h>

int readSeats(int coach[], int N);
int consVac(int coach[], int N);

int main() {

    int coach[72];
    int N;

    printf("Enter the value of N: ");
    scanf("%d", &N);

    readSeats(coach, N);

    printf("The maximum no. of consecutive vacant seats: %d", consVac
(coach, N));
    return 0;
}

int readSeats(int coach[], int N) {
    for (int i = 0; i < N; i++) {
        printf("Enter if the seat is occupied (1) / vacant (0): ");
        scanf("%d", &coach[i]);
    }
}

int consVac(int coach[], int N) {
    int cur = 0, max_count = 0;

    for (int i = 0; i < N; i++) {
        if (coach[i] == 0) {
            cur++;
            if (cur > max_count) {
                max_count = cur;
            }
        } else {
            cur = 0;
        }
    }

    return max_count;
}
```

7. Stock Price – Best Day to Buy & Sell A trader recorded the stock price of a company
   for N consecutive days. He can buy on one day and sell on a later day. Find the
   maximum profit he can achieve. If no profit is possible, print 0. Constraints:
   1 ≤ N ≤ 10^5,
   1 ≤ price[i] ≤ 10^4

```
#include <stdio.h>

int maxProf(int arr[], int N);
void readPrice(int arr[], int N);

int main() {


    int n;
    printf("Enter the no. of consecutive days the trader recorded: ");
    scanf("%d", &n);

    int stocks[n];
    readPrice(stocks, n);

    int mProfit = maxProf(stocks, n);

    printf("\nMaximum Profit: %d", mProfit);
    return 0;
}

void readPrice(int arr[], int N) {
    for (int i = 0; i<N; i++) {
        printf("Enter the price of the stock on day %d: ", i+1);
        scanf("%d", &arr[i]);
    }
}

int maxProf(int arr[], int N) {

    int minPrice = arr[0], maxProfit = 0;

    for (int i = 1; i<N; i++) {
        if (arr[i] < minPrice) {
            minPrice =  arr[i];
        }
        int profit = arr[i] - minPrice;
        if (profit > maxProfit) {
            maxProfit = profit;
        }

    }

    return maxProfit;
}
```

8. Election Vote Counting In a class election, there are N students and each voted for one
   of 5 candidates (candidate numbers 1 to 5). Find the candidate who got the maximum
   votes. If there licant who got the maximum votes. If there is a tie, print "Tie". Constraints:
   1 ≤ N ≤ 1000, 1 ≤ vote[i] ≤ 5

```
#include <stdio.h>

void readVotes(int votes[], int N);
int findWinner(int votes[], int N);

int main() {


    int N;
    printf("Enter number of students: ");
    scanf("%d", &N);

    int votes[N];
    readVotes(votes, N);

    int winner = findWinner(votes, N);
    if (winner == 0)
        printf("Tie");
    else
        printf("Winner is Candidate %d", winner);

    return 0;
}

void readVotes(int votes[], int N) {
    for (int i = 0; i < N; i++) {
        printf("Enter vote of student %d (1-5): ", i + 1);
        scanf("%d", &votes[i]);
    }
}

int findWinner(int votes[], int N) {

    int count[5] = {0};  // count array for candidates 1 to 5

    // Count the votes
    for (int i = 0; i < N; i++) {
        int candidate = votes[i];
        count[candidate - 1]++;   // -1 because array index starts from 0
    }

    int maxVotes = 0, winner = -1;

    // Find candidate with max votes
    for (int i = 0; i < 5; i++) {
        if (count[i] > maxVotes) {
            maxVotes = count[i];
            winner = i + 1;  // candidate number = index + 1
        }
        else if (count[i] == maxVotes && maxVotes != 0) {
            return 0;   // tie found
        }
    }

    return winner;
}
```

9. Traffic Signal Fine System A city installed cameras that record speeds of vehicles for
   the last 7 days. The speed limit is 60 km/h. Count how many times a particular vehicle
   (identified by its number plate string) crossed the speed limit in the week. Constraints: 1
   ≤ N ≤ 1000, 30 ≤ speed ≤ 150

```
#include <stdio.h>

void readSpeeds(int arr[], int days);
int countOverSpeed(int arr[], int days);

int main() {
    char vehicle[20];
    int speeds[7];   // 7 days = fixed size

    printf("Enter the vehicle number plate: ");
    scanf("%s", vehicle);

    printf("\nEnter the speeds for the last 7 days:\n");
    readSpeeds(speeds, 7);

    int count = countOverSpeed(speeds, 7);

    printf("\nVehicle %s crossed the speed limit %d times.\n",
    vehicle, count);

    return 0;
}
```

```
// Function to read speeds
void readSpeeds(int arr[], int days) {
    for (int i = 0; i < days; i++) {
        printf("Day %d speed: ", i + 1);
        scanf("%d", &arr[i]);
    }
}

// Function to count overspeeding occurrences
int countOverSpeed(int arr[], int days) {
    int count = 0;

    for (int i = 0; i < days; i++) {
        if (arr[i] > 60) {   // Speed limit check
            count++;
        }
    }

    return count;
}
```

10. Railway Platform – Minimum Platforms Required Trains arrive and depart at a station
with given arrival and departure times (in 24-hour format as integers, e.g., 930 for 9:30
AM). Find the minimum number of platforms required so that no train waits.
Constraints:
1 ≤ N ≤ 1000, 0000 ≤ time ≤ 2359

```
#include <stdio.h>
#include <stdlib.h>

void readTimes(int arr[], int dep[], int N);
void sort(int arr[], int n);
```

```
int minPlatforms(int arr[], int dep[], int N);

int main() {

    int n;
    printf("Enter number of trains: ");
    scanf("%d", &n);

    int arrival[n], departure[n];

    readTimes(arrival, departure, n);

    int result = minPlatforms(arrival, departure, n);

    printf("\nMinimum Platforms Required: %d\n", result);

    return 0;
}

// Function to read arrival and departure times
void readTimes(int arr[], int dep[], int N) {
    for(int i = 0; i < N; i++) {
        printf("\nTrain %d\n", i + 1);
        printf("Arrival time (e.g., 930): ");
        scanf("%d", &arr[i]);
        printf("Departure time (e.g., 1045): ");
        scanf("%d", &dep[i]);
    }
}

// Simple sort function (bubble sort)
void sort(int arr[], int n) {
    for(int i = 0; i < n-1; i++) {
        for(int j = 0; j < n-i-1; j++) {
            if(arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

// Function to compute minimum required platforms
int minPlatforms(int arr[], int dep[], int N) {

    sort(arr, N);
    sort(dep, N);

    int i = 0, j = 0;
    int platforms = 0, maxPlatforms = 0;

    while(i < N && j < N) {

        if(arr[i] <= dep[j]) {
            // A train has arrived → need a platform
            platforms++;
            if(platforms > maxPlatforms)
                maxPlatforms = platforms;
            i++;
        }
        else {
            // A train has departed → free a platform
            platforms--;
            j++;
        }
    }

    return maxPlatforms;
}
```

▼ 2D Arrays / Matrices

11. Movie Theater Seat Booking A multiplex has a 10×10 seating arrangement. Booked
seats are marked 'B', available are 'A'. The manager wants to print the seating chart and
also tell how many prime-numbered seats (row×10 + col is prime) are still available.
Constraints: Fixed 10×10 grid

```
#include <stdio.h>
#include <ctype.h>

#define ROWS 10
#define COLS 10

void readSeating(char seats[ROWS][COLS]);
void printSeating(char seats[ROWS][COLS]);
int isPrime(int n);
int countPrimeAvailable(char seats[ROWS][COLS]);

int main() {

    char seats[ROWS][COLS];

    printf("Enter seating chart (A for available, B for booked):\n");
    readSeating(seats);

    printf("\n--- Seating Chart ---\n");
    printSeating(seats);

    int primeAvailable = countPrimeAvailable(seats);

    printf("\nPrime-numbered seats available: %d\n", primeAvailable);

    return 0;
}
```

```
// Reads the 10×10 seating chart
void readSeating(char seats[ROWS][COLS]) {
    for(int i = 0; i < ROWS; i++) {
        for(int j = 0; j < COLS; j++) {
            scanf(" %c", &seats[i][j]);   // space before %c to ignore whites
pace
            seats[i][j] = toupper(seats[i][j]);
        }
    }
}

// Prints seating chart
void printSeating(char seats[ROWS][COLS]) {
    for(int i = 0; i < ROWS; i++) {
        for(int j = 0; j < COLS; j++) {
            printf("%c ", seats[i][j]);
        }
        printf("\n");
    }
}

// Prime check function
int isPrime(int n) {
    if (n <= 1) return 0;
    for(int i = 2; i * i <= n; i++) {
        if(n % i == 0) return 0;
    }
    return 1;
}

// Count available seats that are prime-numbered
int countPrimeAvailable(char seats[ROWS][COLS]) {
    int count = 0;

    for(int i = 0; i < ROWS; i++) {
        for(int j = 0; j < COLS; j++) {
            int seatNum = i * 10 + j;   // seat number rule

            if(seats[i][j] == 'A' && isPrime(seatNum)) {
                count++;
            }
        }
    }

    return count;
}
```

12. Treasure Hunt Game An archaeologist found a 5×5 grid where each cell contains a digit
(0–9). A treasure is located where the 3×3 subgrid (centered on that cell) has the
maximum sum. Find the maximum possible sum. Constraints: 5×5 grid, 0 ≤ digit ≤ 9

```
#include <stdio.h>

#define SIZE 5

void readGrid(int grid[SIZE][SIZE]);
int max3×3Sum(int grid[SIZE][SIZE]);

int main() {
    int grid[SIZE][SIZE];

    printf("Enter the 5×5 grid values (0–9):\n");
    readGrid(grid);

    int result = max3×3Sum(grid);

    printf("\nMaximum 3×3 Subgrid Sum: %d\n", result);
    return 0;
}

// Function to read the 5×5 grid from user
void readGrid(int grid[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            scanf("%d", &grid[i][j]);
        }
    }
}

// Function to compute maximum 3×3 subgrid sum
int max3×3Sum(int grid[SIZE][SIZE]) {
    int maxSum = 0;

    // The center of a 3×3 grid can only be at positions (1,1) to (3,3)
    for (int i = 1; i < SIZE - 1; i++) {
        for (int j = 1; j < SIZE - 1; j++) {

            int sum = 0;
            // Loop over 3×3 region centered at (i, j)
            for (int r = i - 1; r <= i + 1; r++) {
                for (int c = j - 1; c <= j + 1; c++) {
                    sum += grid[r][c];
                }
            }

            if (sum > maxSum)
                maxSum = sum;
        }
    }
    return maxSum;
}
```

13. Bank Locker System A bank has a 9×9 locker matrix. Lockers are numbered 1 to 81.
    The bank wants to print all locker numbers on the main diagonal and anti-diagonal (like
    X shape) in sorted order. Constraints: 9×9 fixed grid

```
#include <stdio.h>

#define SIZE 9

void fillLockers(int grid[SIZE][SIZE]);
void getDiagonalLockers(int grid[SIZE][SIZE], int result[], int *count);
void sortArray(int arr[], int n);
void printArray(int arr[], int n);

int main() {

    int grid[SIZE][SIZE];
    int diagLockers[2 * SIZE];
    int count = 0;

    fillLockers(grid);
    getDiagonalLockers(grid, diagLockers, &count);

    sortArray(diagLockers, count);

    printf("Diagonal Locker Numbers in Sorted Order:\n");
    printArray(diagLockers, count);

    return 0;
}
```

```
void fillLockers(int grid[SIZE][SIZE]) {
    int num = 1;
    for (int i = 0; i < SIZE; i++)
        for (int j = 0; j < SIZE; j++)
            grid[i][j] = num++;
}

void getDiagonalLockers(int grid[SIZE][SIZE], int result[], int *count) {

    // Main diagonal (i == j)
    for (int i = 0; i < SIZE; i++) {
        result[(*count)++] = grid[i][i];
    }

    // Anti-diagonal (i + j == SIZE - 1)
    for (int i = 0; i < SIZE; i++) {
        int j = SIZE - 1 - i;

        // Avoid duplicate center element (i == j at center)
        if (i != j) {
            result[(*count)++] = grid[i][j];
        }
    }
}

void sortArray(int arr[], int n) {
    for (int i = 0; i < n - 1; i++)
        for (int j = i + 1; j < n; j++)
            if (arr[i] > arr[j]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
}
```

▼ Searching & Sorting

14. Library Book Search The library has N books sorted by their ISBN number. A student
wants to borrow a book with a specific ISBN. Write a program using binary search to
return the position (1-based index) if found, otherwise return -1.
Constraints: 1 ≤ N ≤
10^5, ISBNs are unique and sorted

```
#include <stdio.h>

// Function to perform binary search
int binarySearch(int arr[], int n, int key) {
    int left = 0, right = n - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == key)
            return mid + 1;  // return 1-based index

        else if (arr[mid] < key)
            left = mid + 1;

        else
            right = mid - 1;
    }

    return -1;  // not found
}
```

```
int main() {
    int n, key;

    printf("Enter number of books: ");
    scanf("%d", &n);

    int books[n];

    printf("Enter %d sorted ISBN numbers:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &books[i]);
    }

    printf("Enter ISBN to search: ");
    scanf("%d", &key);

    int position = binarySearch(books, n, key);

    if (position == -1)
        printf("Book not found.\n");
    else
        printf("Book found at position %d\n", position);

    return 0;
}
```

15. Hospital Patient Priority Queue Patients arrive with severity levels (1 to 100). The
hospital uses Selection Sort to always treat the most critical patient next. Simulate the
sorting of the first 10 patients and print their severity in decreasing order. Constraints: 1 ≤
N ≤ 100, 1 ≤ severity ≤ 100

```
#include <stdio.h>

// Function to perform selection sort in decreasing order
void selectionSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int maxIndex = i;

        for (int j = i + 1; j < n; j++) {
            if (arr[j] > arr[maxIndex]) {
                maxIndex = j;
            }
        }

        // Swap
        int temp = arr[i];
        arr[i] = arr[maxIndex];
        arr[maxIndex] = temp;
    }
}

int main() {
    int N;
    printf("Enter number of patients (N ≤ 100): ");
    scanf("%d", &N);

    int patients[N];

    printf("Enter severity levels (1 to 100):\n");
    for (int i = 0; i < N; i++) {
        scanf("%d", &patients[i]);
    }

    int limit = (N < 10) ? N : 10;  // only sort first 10 patients

    selectionSort(patients, limit);
    printf("\nTop %d patients in decreasing severity:\n", limit);
    for (int i = 0; i < limit; i++) {
        printf("%d ", patients[i]);
    }

    return 0;
}
```

▼ Mixed Popular Problems

16. Rainwater Harvesting in Colony There are N buildings in a row with different heights.
    After rainfall, water gets trapped between buildings. Calculate the total units of water that
    can be trapped. Constraints: 1 ≤ N ≤ 1000, 0 ≤ height[i] ≤ 10^5

```
#include <stdio.h>

// Function to compute total trapped water
int calculateTrappedWater(int height[], int n) {
    if (n <= 2) return 0;  // No water can be trapped

    int leftMax[n], rightMax[n];

    // Compute left max boundary for each building
    leftMax[0] = height[0];
    for (int i = 1; i < n; i++) {
        leftMax[i] = (height[i] > leftMax[i - 1]) ? height[i] : leftMax[i - 1];
    }

    // Compute right max boundary for each building
    rightMax[n - 1] = height[n - 1];
    for (int i = n - 2; i >= 0; i--) {
        rightMax[i] = (height[i] > rightMax[i + 1]) ?
                                height[i] : rightMax[i + 1];
    }

    // Calculate trapped water
    int totalWater = 0;
    for (int i = 0; i < n; i++) {
        int trap = (leftMax[i] < rightMax[i] ? leftMax[i] : rightMax[i]) -
                        height[i];
        totalWater += trap;
    }

    return totalWater;
}

int main() {
    int n;
    printf("Enter number of buildings (N ≤ 1000): ");
    scanf("%d", &n);

    int height[n];
    printf("Enter heights of buildings:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &height[i]);
    }

    int trappedWater = calculateTrappedWater(height, n);

    printf("Total trapped rainwater: %d units\n", trappedWater);

    return 0;
}
```

17. Flipkart Delivery – Maximum Packages in One Trip A delivery boy can carry a
maximum weight W. There are N packages with given weights. Find the maximum

number of packages he can carry in one trip without exceeding weight W.
Constraints: 1
≤ N ≤ 100, 1 ≤ weight[i] ≤ 1000, 1 ≤ W ≤ 10000.

```
#include <stdio.h>

// Function to sort package weights in increasing order (Selection Sort)
void selectionSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;

        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }

        // Swap
        int temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
    }
}

// Function to calculate maximum packages that can be carried
int maxPackages(int weights[], int n, int W) {
    int count = 0;
    int currentWeight = 0;

    for (int i = 0; i < n; i++) {
        if (currentWeight + weights[i] <= W) {
            currentWeight += weights[i];
            count++;
        } else {
            break;
        }
    }

    return count;
}

int main() {
    int n, W;

    printf("Enter number of packages (N ≤ 100): ");
    scanf("%d", &n);

    int weights[n];

    printf("Enter weights of %d packages:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &weights[i]);
    }

    printf("Enter maximum weight delivery boy can carry (W ≤ 10000):
");
    scanf("%d", &W);

    // Sort the weights first
    selectionSort(weights, n);

    // Calculate maximum packages
    int result = maxPackages(weights, n, W);

    printf("Maximum number of packages he can carry: %d\n", result);

    return 0;
}
```