

1. Sum of Elements in an Array

- Scenario: You have a list of daily sales of a store.
- a) Write a simple iterative program to compute the sum of elements.
- b) Write a recursive version of the same program.

```
main.c
1 #include <stdio.h>
2 int main() {
3     int n, i;
4     printf("Enter number of days: ");
5     scanf("%d", &n);
6     int sales[n];
7     printf("Enter daily sales:\n");
8     for (i = 0; i < n; i++) {
9         scanf("%d", &sales[i]);
10    }
11    int total = 0;
12    for (i = 0; i < n; i++) {
13        total += sales[i];
14    }
15    printf("Total sales (Iterative): %d\n", total);
16    return 0;
17 }
18
```

input

Enter number of days: 5
Enter daily sales:
120 132 135 140 145
Total sales (Iterative): 672

```
main.c
1 #include <stdio.h>
2 int sum_recursive(int arr[], int n) {
3     if (n == 0)
4         return 0;
5     else
6         return arr[n - 1] + sum_recursive(arr, n - 1);
7 }
8 int main() {
9     int n, i;
10    printf("Enter number of days: ");
11    scanf("%d", &n);
12
13    int sales[n];
14    printf("Enter daily sales:\n");
15    for (i = 0; i < n; i++) {
16        scanf("%d", &sales[i]);
17    }
18    int total = sum_recursive(sales, n);
19    printf("Total sales (Recursive): %d\n", total);
20    return 0;
21 }
22
```

input

Enter number of days: 5
Enter daily sales:
120 125 132 118 130
Total sales (Recursive): 625

2. Find Maximum Element in an Array

- Scenario: Given temperature readings for a week.

a) Find maximum element using a simple linear scan ($O(n)$).

b) Can you find it by first sorting the array and taking the last element?

```
main.c
1 #include <stdio.h>
2 int main() {
3     int n, i;
4     printf("Enter number of days: ");
5     scanf("%d", &n);
6     int temp[n];
7     printf("Enter temperature readings:\n");
8     for (i = 0; i < n; i++) {
9         scanf("%d", &temp[i]);
10    }
11    int max = temp[0];
12    for (i = 1; i < n; i++) {
13        if (temp[i] > max) {
14            max = temp[i];
15        }
16    }
17    printf("Maximum temperature (Linear Scan): %d\n", max);
18    return 0;
19 }
20
```

input

Enter number of days: 5
Enter temperature readings:
100 101 99 102 100
Maximum temperature (Linear Scan): 102

```
main.c
1 #include <stdio.h>
2 void bubbleSort(int arr[], int n) {
3     int i, j, temp;
4     for (i = 0; i < n - 1; i++) {
5         for (j = 0; j < n - i - 1; j++) {
6             if (arr[j] > arr[j + 1]) {
7                 temp = arr[j];
8                 arr[j] = arr[j + 1];
9                 arr[j + 1] = temp;
10            }
11        }
12    }
13 }
14 int main() {
15     int n, i;
16     printf("Enter number of days: ");
17     scanf("%d", &n);
18     int temp[n];
19     printf("Enter temperature readings:\n");
20     for (i = 0; i < n; i++) {
21         scanf("%d", &temp[i]);
22     }
23     bubbleSort(temp, n);
24     int max = temp[n - 1];
25     printf("Maximum temperature (After Sorting): %d\n", max);
26     return 0;
27 }
28
```

input

Enter number of days: 5
Enter temperature readings:
100 101 99 102 101
Maximum temperature (After Sorting): 102

3. Reverse a String

- Scenario: Reversing a username for some encryption purpose.
- a) Reverse in-place using two-pointer technique ($O(n)$, $O(1)$).
- b) Reverse by creating a new array ($O(n)$, $O(n)$).

```
main.c
1 #include <stdio.h>
2 #include <string.h>
3 int main() {
4     char username[100];
5     printf("Enter username: ");
6     scanf("%s", username);
7     int start = 0;
8     int end = strlen(username) - 1;
9     char temp;
10    while (start < end) {
11        temp = username[start];
12        username[start] = username[end];
13        username[end] = temp;
14        start++;
15        end--;
16    }
17    printf("Reversed username (in-place): %s\n", username);
18    return 0;
19 }
20
```

input

Enter username: atul dhiman.ad
Reversed username (in-place): da.namihd_luta

```
main.c
1 #include <stdio.h>
2 #include <string.h>
3 int main() {
4     char username[100];
5     printf("Enter username: ");
6     scanf("%s", username);
7     int len = strlen(username);
8     char reversed[100];
9     for (int i = 0; i < len; i++) {
10        reversed[i] = username[len - i - 1];
11    }
12    reversed[len] = '\0';
13    printf("Reversed username (new array): %s\n", reversed);
14    return 0;
15 }
16
```

input

Enter username: atul dhiman.ad
Reversed username (new array): da.namihd_luta

4. Check Even or Odd

- Scenario: Given a list of numbers, print whether each number is even or odd.

a) Using modulo operator ($O(1)$).

b) Using bitwise AND ($n \& 1$) ($O(1)$).

```
main.c
1 #include <stdio.h>
2 int main() {
3     int n;
4     printf("Enter number of elements: ");
5     scanf("%d", &n);
6     int arr[n];
7     printf("Enter numbers:\n");
8     for (int i = 0; i < n; i++) {
9         scanf("%d", &arr[i]);
10    }
11    printf("\nChecking using modulo operator:\n");
12    for (int i = 0; i < n; i++) {
13        if (arr[i] % 2 == 0)
14            printf("%d is Even\n", arr[i]);
15        else
16            printf("%d is Odd\n", arr[i]);
17    }
18    return 0;
19 }
20
```

input

Enter number of elements: 5
Enter numbers:
7 60 45 69 88

Checking using modulo operator:
7 is Odd
60 is Even
45 is Odd
69 is Odd
88 is Even

```
main.c
1 #include <stdio.h>
2 int main() {
3     int n;
4     printf("Enter number of elements: ");
5     scanf("%d", &n);
6     int arr[n];
7     printf("Enter numbers:\n");
8     for (int i = 0; i < n; i++) {
9         scanf("%d", &arr[i]);
10    }
11    printf("\nChecking using bitwise AND:\n");
12    for (int i = 0; i < n; i++) {
13        if ((arr[i] & 1) == 0)
14            printf("%d is Even\n", arr[i]);
15        else
16            printf("%d is Odd\n", arr[i]);
17    }
18    return 0;
19 }
20
```

input

Enter number of elements: 5
Enter numbers:
12 43 55 22 14

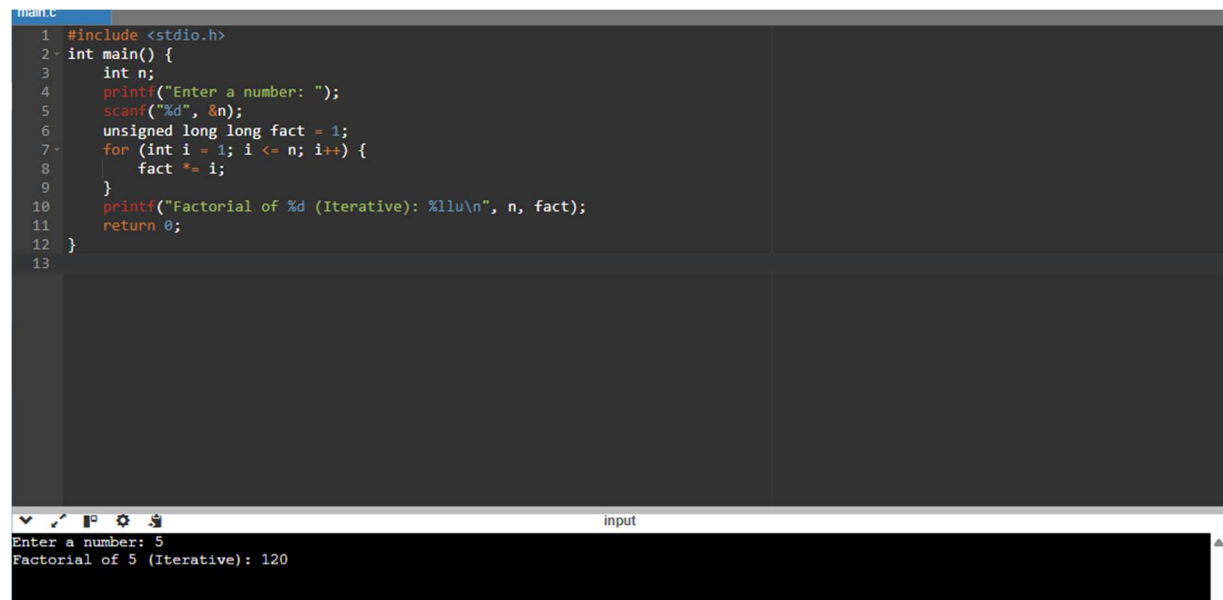
Checking using bitwise AND:
12 is Even
43 is Odd
55 is Odd
22 is Even
14 is Even

5. Factorial of a Number

- Scenario: Calculate the number of possible arrangements.

a) Iterative method ($O(n)$, $O(1)$).

b) Recursive method ($O(n)$, $O(n)$ for call stack).



```
main.c
1 #include <stdio.h>
2 int main() {
3     int n;
4     printf("Enter a number: ");
5     scanf("%d", &n);
6     unsigned long long fact = 1;
7     for (int i = 1; i <= n; i++) {
8         fact *= i;
9     }
10    printf("Factorial of %d (Iterative): %llu\n", n, fact);
11    return 0;
12 }
13
```

input

Enter a number: 5
Factorial of 5 (Iterative): 120



```
main.c
1 #include <stdio.h>
2 unsigned long long factorial(int n) {
3     if (n == 0 || n == 1)
4         return 1;
5     else
6         return n * factorial(n - 1);
7 }
8 int main() {
9     int n;
10    printf("Enter a number: ");
11    scanf("%d", &n);
12
13    printf("Factorial of %d (Recursive): %llu\n", n, factorial(n));
14    return 0;
15 }
16
```

Language C

input


Enter a number: 5
Factorial of 5 (Recursive): 120

6. Linear Search

- Scenario: Search for a customer ID in a small dataset.

a) Implement basic linear search ($O(n)$, $O(1)$).

b) Optimize by using sentinel method to reduce comparisons (still $O(n)$, but fewer operations).

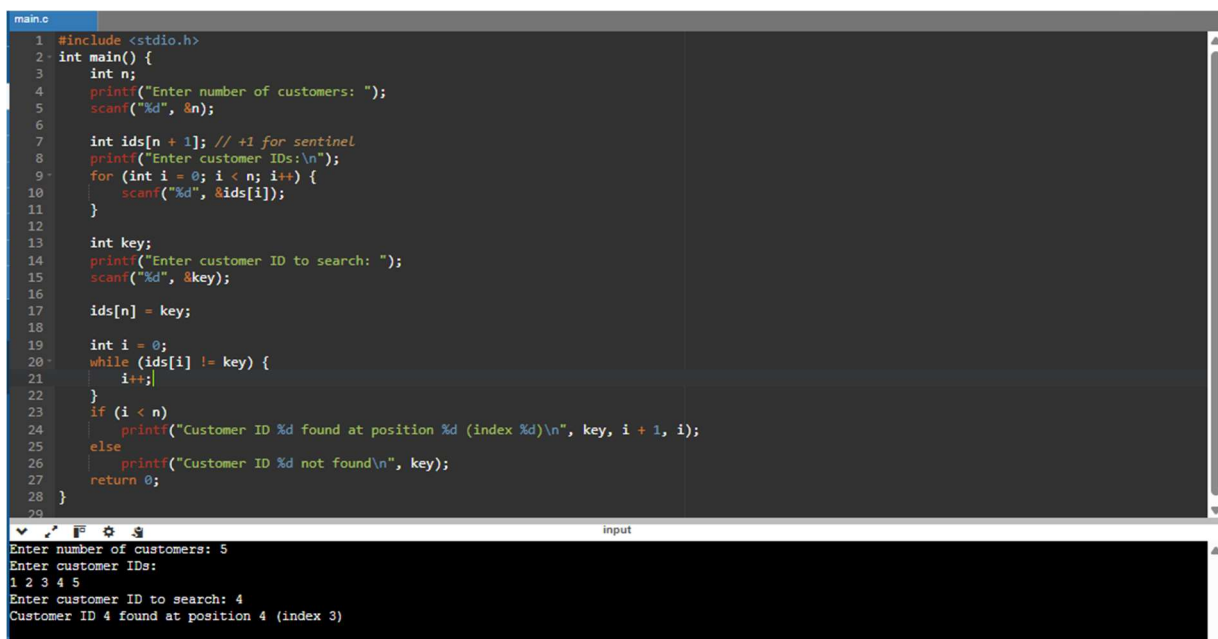


The screenshot shows a C program in a code editor. The program prompts the user to enter the number of customers, then the customer IDs, and finally the customer ID to search for. It uses a for loop to iterate through the IDs and a break statement to exit the loop once the search key is found. The output shows the search for ID 4 at index 3.

```
1 #include <stdio.h>
2 int main() {
3     int n;
4     printf("Enter number of customers: ");
5     scanf("%d", &n);
6     int ids[n];
7     printf("Enter customer IDs:\n");
8     for (int i = 0; i < n; i++) {
9         scanf("%d", &ids[i]);
10    }
11    int key;
12    printf("Enter customer ID to search: ");
13    scanf("%d", &key);
14    int found = -1;
15    for (int i = 0; i < n; i++) {
16        if (ids[i] == key) {
17            found = i;
18            break;
19        }
20    }
21    if (found != -1)
22        printf("Customer ID %d found at position %d (index %d)\n", key, found + 1, found);
23    else
24        printf("Customer ID %d not found\n", key);
25    return 0;
26 }
```

input

Enter number of customers: 5
Enter customer IDs:
1 2 3 4 5
Enter customer ID to search: 4
Customer ID 4 found at position 4 (index 3)



The screenshot shows a C program in a code editor. This version uses a sentinel value to optimize the linear search. It adds an extra element to the array and sets it to the search key. The search is performed using a while loop that continues until the key is found or the end of the array is reached. The output is the same as the first program.

```
1 #include <stdio.h>
2 int main() {
3     int n;
4     printf("Enter number of customers: ");
5     scanf("%d", &n);
6
7     int ids[n + 1]; // +1 for sentinel
8     printf("Enter customer IDs:\n");
9     for (int i = 0; i < n; i++) {
10        scanf("%d", &ids[i]);
11    }
12
13    int key;
14    printf("Enter customer ID to search: ");
15    scanf("%d", &key);
16
17    ids[n] = key;
18
19    int i = 0;
20    while (ids[i] != key) {
21        i++;
22    }
23    if (i < n)
24        printf("Customer ID %d found at position %d (index %d)\n", key, i + 1, i);
25    else
26        printf("Customer ID %d not found\n", key);
27    return 0;
28 }
```

input

Enter number of customers: 5
Enter customer IDs:
1 2 3 4 5
Enter customer ID to search: 4
Customer ID 4 found at position 4 (index 3)

7. Print First n Natural Numbers

- Scenario: Generate first n natural numbers for a report.

a) Using a simple for loop ($O(n)$, $O(1)$).

b) Using recursion ($O(n)$, $O(n)$).

```
main.c
1 #include <stdio.h>
2 int main() {
3     int n;
4     printf("Enter n: ");
5     scanf("%d", &n);
6     printf("First %d natural numbers (loop):\n", n);
7     for (int i = 1; i <= n; i++) {
8         printf("%d ", i);
9     }
10    printf("\n");
11    return 0;
12 }
13
```

input

Enter n: 5
First 5 natural numbers (loop):
1 2 3 4 5

```
main.c
1 #include <stdio.h>
2 void printNumbers(int current, int n) {
3     if (current > n)
4         return;
5     printf("%d ", current);
6     printNumbers(current + 1, n);
7 }
8 int main() {
9     int n;
10    printf("Enter n: ");
11    scanf("%d", &n);
12    printf("First %d natural numbers (recursion):\n", n);
13    printNumbers(1, n);
14    printf("\n");
15    return 0;
16 }
17
```

input

Enter n: 5
First 5 natural numbers (recursion):
1 2 3 4 5

8. Count Vowels in a String

- Scenario: Analyze user comments to count number of vowels.

a) Traverse string and check each character ($O(n)$, $O(1)$).

b) Use a lookup table (array of 256 size) to speed up vowel checking ($O(n)$, $O(1)$ but extra space).

```
main.c
1 #include <stdio.h>
2 #include <string.h>
3 #include <ctype.h>
4 int main() {
5     char comment[500];
6     printf("Enter user comment: ");
7     fgets(comment, sizeof(comment), stdin);
8     int count = 0;
9     for (int i = 0; i < strlen(comment); i++) {
10         char c = tolower(comment[i]);
11         if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {
12             count++;
13         }
14     }
15     printf("Number of vowels (simple check): %d\n", count);
16     return 0;
17 }
18
```

input

Enter user comment: Atul Dhiman Artificial Intelligence
Number of vowels (simple check): 14

```
main.c
1 #include <stdio.h>
2 #include <string.h>
3 int main() {
4     char comment[500];
5     printf("Enter user comment: ");
6     fgets(comment, sizeof(comment), stdin);
7     int isVowel[256] = {0};
8     isVowel['a'] = 1; isVowel['e'] = 1; isVowel['i'] = 1; isVowel['o'] = 1; isVowel['u'] = 1;
9     isVowel['A'] = 1; isVowel['E'] = 1; isVowel['I'] = 1; isVowel['O'] = 1; isVowel['U'] = 1;
10     int count = 0;
11     for (int i = 0; i < strlen(comment); i++) {
12         unsigned char c = comment[i];
13         if (isVowel[c]) {
14             count++;
15         }
16     }
17     printf("Number of vowels (lookup table): %d\n", count);
18     return 0;
19 }
20
```

input

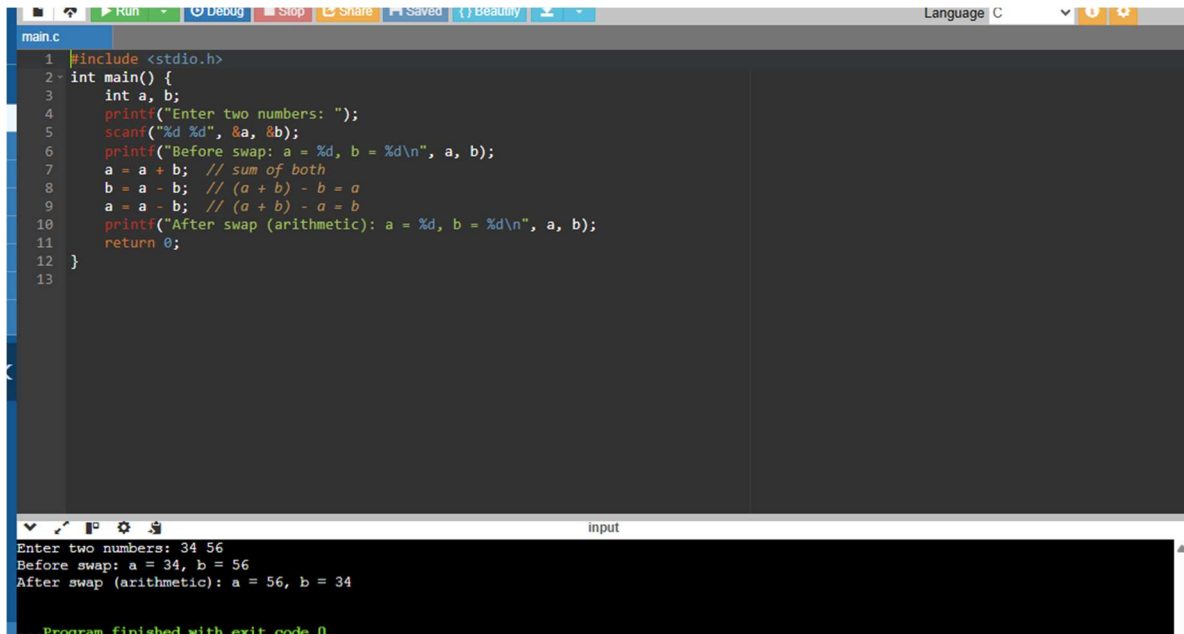
Enter user comment: Atul Dhiman Artificial Intelligence
Number of vowels (lookup table): 14

9. Swap Two Numbers Without Temporary Variable

- Scenario: In embedded systems where memory is constrained.

a) Using arithmetic ($a = a + b$; $b = a - b$; $a = a - b$).

b) Using bitwise XOR ($a = a \oplus b$; $b = a \oplus b$; $a = a \oplus b$).

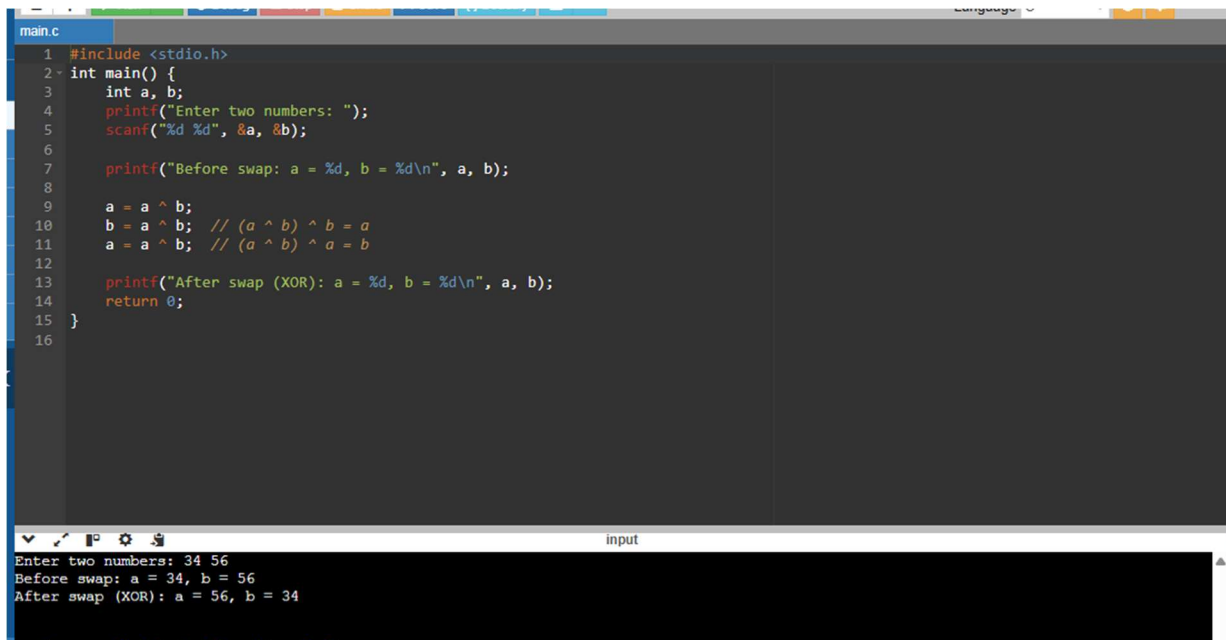


The screenshot shows a C program in a code editor. The code uses arithmetic to swap two numbers. The output window shows the program's execution with input values 34 and 56, and the resulting swapped values 56 and 34.

```
main.c
1 #include <stdio.h>
2 int main() {
3     int a, b;
4     printf("Enter two numbers: ");
5     scanf("%d %d", &a, &b);
6     printf("Before swap: a = %d, b = %d\n", a, b);
7     a = a + b; // sum of both
8     b = a - b; // (a + b) - b = a
9     a = a - b; // (a + b) - a = b
10    printf("After swap (arithmetic): a = %d, b = %d\n", a, b);
11    return 0;
12 }
13
```

input

Enter two numbers: 34 56
Before swap: a = 34, b = 56
After swap (arithmetic): a = 56, b = 34
Program finished with exit code 0



The screenshot shows a C program in a code editor. The code uses bitwise XOR to swap two numbers. The output window shows the program's execution with input values 34 and 56, and the resulting swapped values 56 and 34.

```
main.c
1 #include <stdio.h>
2 int main() {
3     int a, b;
4     printf("Enter two numbers: ");
5     scanf("%d %d", &a, &b);
6
7     printf("Before swap: a = %d, b = %d\n", a, b);
8
9     a = a ^ b;
10    b = a ^ b; // (a ^ b) ^ b = a
11    a = a ^ b; // (a ^ b) ^ a = b
12
13    printf("After swap (XOR): a = %d, b = %d\n", a, b);
14    return 0;
15 }
16
```

input

Enter two numbers: 34 56
Before swap: a = 34, b = 56
After swap (XOR): a = 56, b = 34
Program finished with exit code 0

10. Check Palindrome Number

- Scenario: Validate identification number (ID) is symmetric.

a) Convert number to string and check ($O(n)$, $O(n)$).

b) Mathematical method: Reverse digits without converting ($O(\log n)$, $O(1)$).

The screenshot shows the OnlineGDB interface with a C program for checking a palindrome using string conversion. The program is named 'main.c' and is located at 'onlinegdb.com/edit/6llu85fE'. The code is as follows:

```
1 #include <stdio.h>
2 #include <string.h>
3 int main() {
4     char id[50];
5     printf("Enter ID number: ");
6     scanf("%s", id);
7     int len = strlen(id);
8     int isPalindrome = 1;
9     for (int i = 0; i < len / 2; i++) {
10         if (id[i] != id[len - i - 1]) {
11             isPalindrome = 0;
12             break;
13         }
14     }
15     if (isPalindrome)
16         printf("The ID %s is a palindrome (string check)\n", id);
17     else
18         printf("The ID %s is NOT a palindrome (string check)\n", id);
19     return 0;
20 }
```

The output shows the user entering '110011' and the program outputting 'The ID 110011 is a palindrome (string check)'. The program finished with exit code 0.

The screenshot shows the OnlineGDB interface with a C program for checking a palindrome using mathematical reversal. The program is named 'main.c' and is located at 'onlinegdb.com/edit/q9JOFJLWG_'. The code is as follows:

```
1 #include <stdio.h>
2 int main() {
3     long long num;
4     printf("Enter ID number: ");
5     scanf("%lld", &num);
6     long long original = num;
7     long long reversed = 0;
8     while (num > 0) {
9         int digit = num % 10;
10        reversed = reversed * 10 + digit;
11        num = num / 10;
12    }
13    if (original == reversed)
14        printf("The ID %lld is a palindrome (math check)\n", original);
15    else
16        printf("The ID %lld is NOT a palindrome (math check)\n", original);
17    return 0;
18 }
```

The output shows the user entering '11012' and the program outputting 'The ID 11012 is NOT a palindrome (math check)'. The program finished with exit code 0.