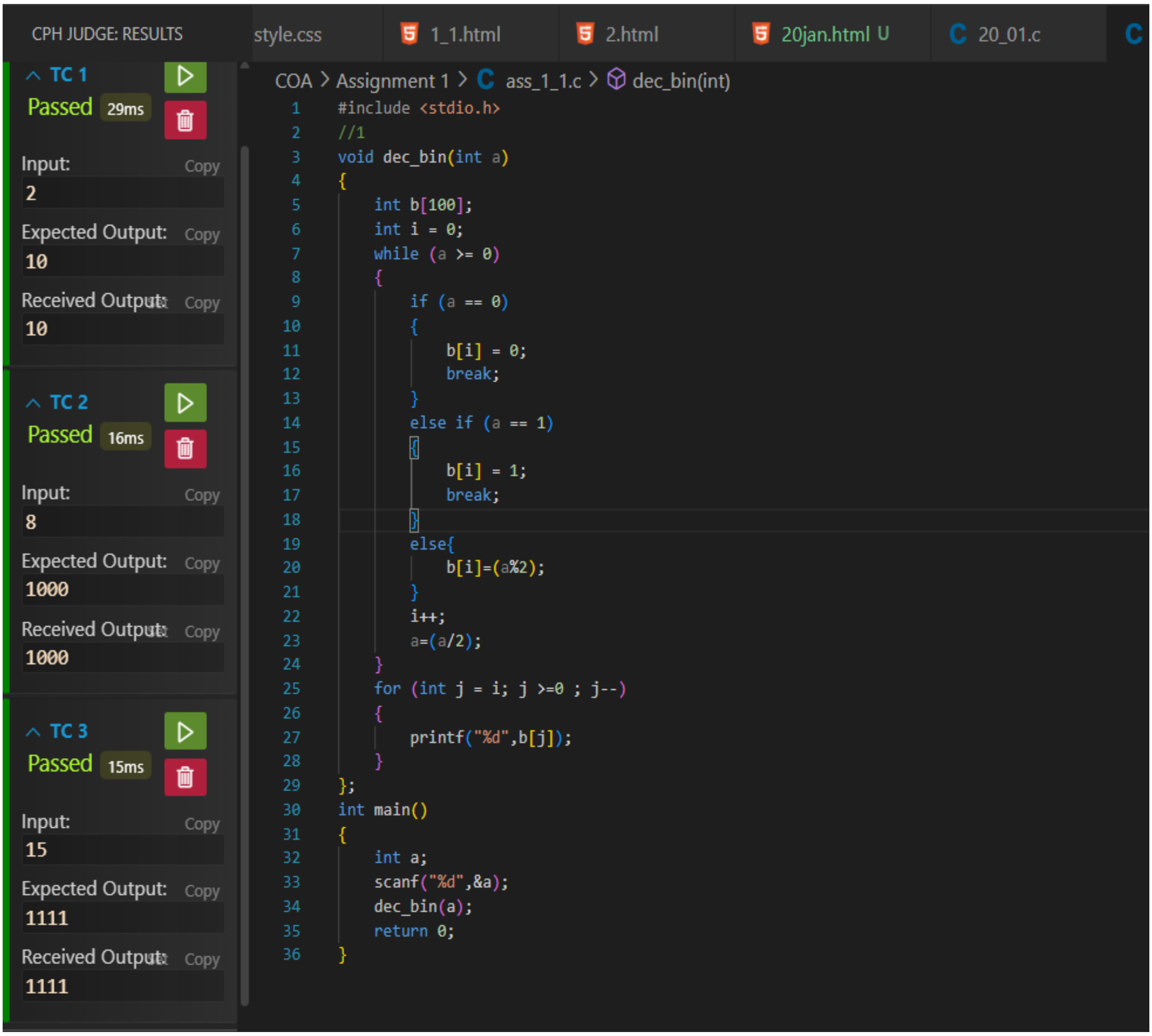
Q1



TC 1
Passed 29ms

Input: Copy
```
2
```
Expected Output: Copy
```
10
```
Received Output: Copy
```
10
```

TC 2
Passed 16ms

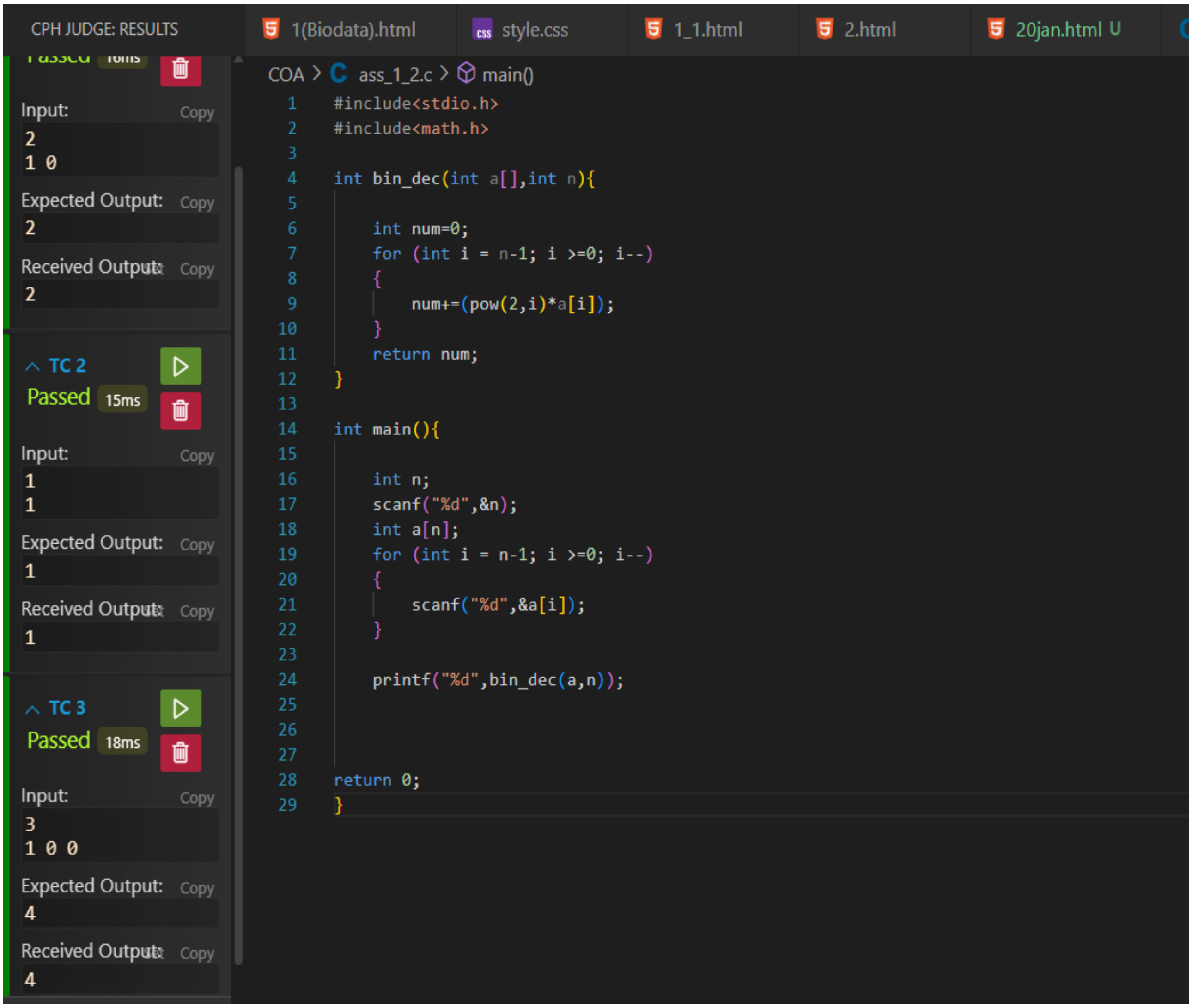
Input: Copy
```
8
```
Expected Output: Copy
```
1000
```
Received Output: Copy
```
1000
```

TC 3
Passed 15ms

Input: Copy
```
15
```
Expected Output: Copy
```
1111
```
Received Output: Copy
```
1111
```

```c
#include <stdio.h>
//1
void dec_bin(int a)
{
    int b[100];
    int i = 0;
    while (a >= 0)
    {
        if (a == 0)
        {
            b[i] = 0;
            break;
        }
        else if (a == 1)
        {
            b[i] = 1;
            break;
        }
        else{
            b[i]=(a%2);
        }
        i++;
        a=(a/2);
    }
    for (int j = i; j >=0 ; j--)
    {
        printf("%d",b[j]);
    }
};
int main()
{
    int a;
    scanf("%d",&a);
    dec_bin(a);
    return 0;
}
```

Q2



Passed 16ms

Input:
```
2
1 0
```
Expected Output:
```
2
```
Received Output:
```
2
```

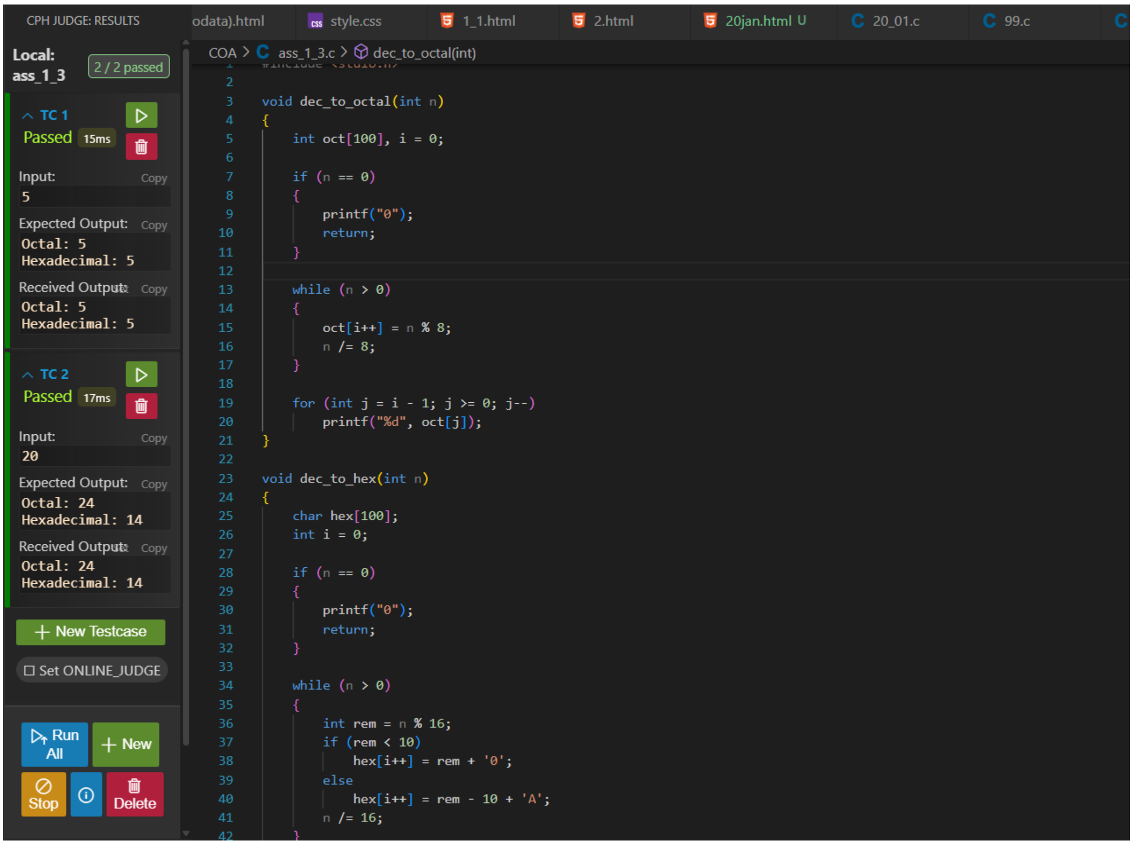
TC 2
Passed 15ms

Input:
```
1
1
```
Expected Output:
```
1
```
Received Output:
```
1
```

TC 3
Passed 18ms

Input:
```
3
1 0 0
```
Expected Output:
```
4
```
Received Output:
```
4
```

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

int bin_dec(int a[],int n){

    int num=0;
    for (int i = n-1; i >=0; i--)
    {
        num+=(pow(2,i)*a[i]);
    }
    return num;
}

int main(){

    int n;
    scanf("%d",&n);
    int a[n];
    for (int i = n-1; i >=0; i--)
    {
        scanf("%d",&a[i]);
    }

    printf("%d",bin_dec(a,n));



return 0;
}
```



Local:
ass_1_3
2 / 2 passed

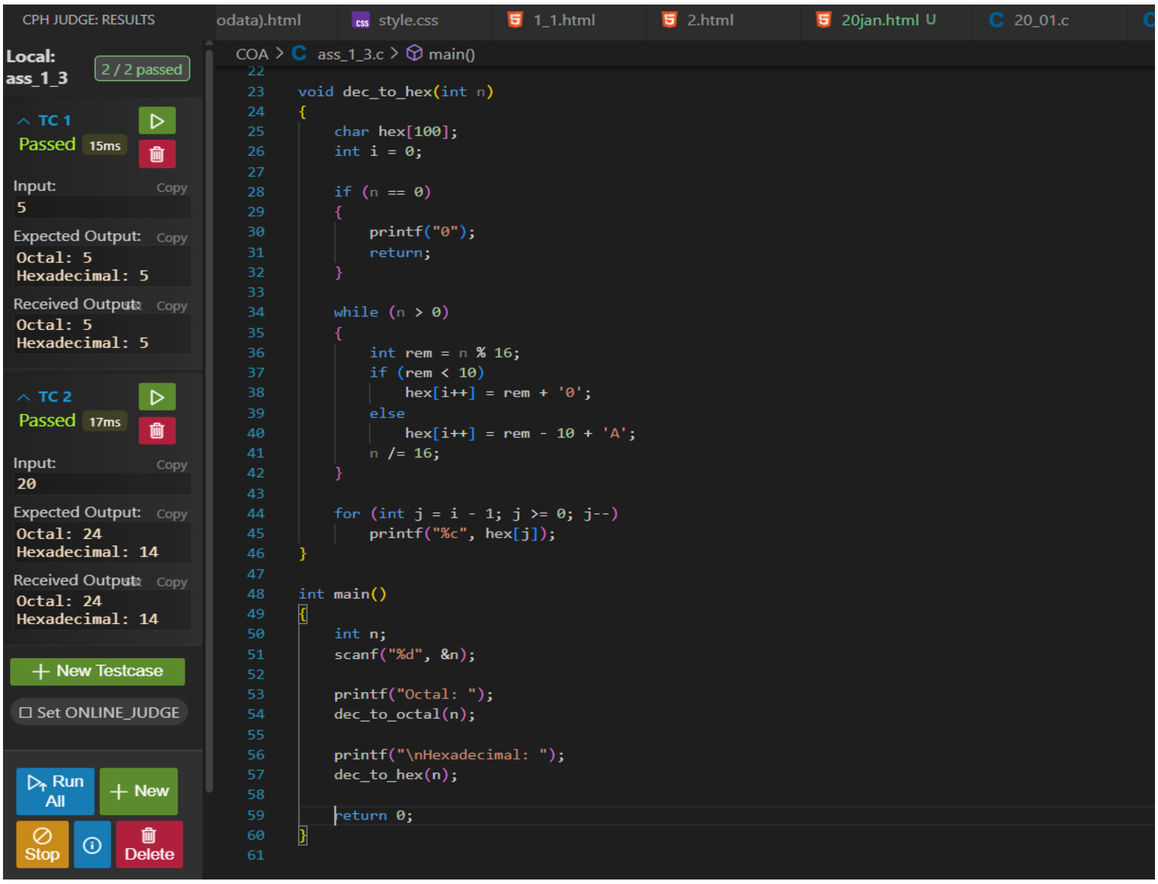
TC 1
Passed 15ms

Input: Copy
```
5
```
Expected Output: Copy
```
Octal: 5
Hexadecimal: 5
```
Received Output: Copy
```
Octal: 5
Hexadecimal: 5
```

TC 2
Passed 17ms

Input: Copy
```
20
```
Expected Output: Copy
```
Octal: 24
Hexadecimal: 14
```
Received Output: Copy
```
Octal: 24
Hexadecimal: 14
```

```c
#include <stdio.h>

void dec_to_octal(int n)
{
    int oct[100], i = 0;

    if (n == 0)
    {
        printf("0");
        return;
    }

    while (n > 0)
    {
        oct[i++] = n % 8;
        n /= 8;
    }

    for (int j = i - 1; j >= 0; j--)
        printf("%d", oct[j]);
}

void dec_to_hex(int n)
{
    char hex[100];
    int i = 0;

    if (n == 0)
    {
        printf("0");
        return;
    }

    while (n > 0)
    {
        int rem = n % 16;
        if (rem < 10)
            hex[i++] = rem + '0';
        else
            hex[i++] = rem - 10 + 'A';
        n /= 16;
    }
```



Local:
ass_1_3
2 / 2 passed

TC 1
Passed 15ms

Input: Copy
5
Expected Output: Copy
Octal: 5
Hexadecimal: 5
Received Output: Copy
Octal: 5
Hexadecimal: 5

TC 2
Passed 17ms

Input: Copy
20
Expected Output: Copy
Octal: 24
Hexadecimal: 14
Received Output: Copy
Octal: 24
Hexadecimal: 14

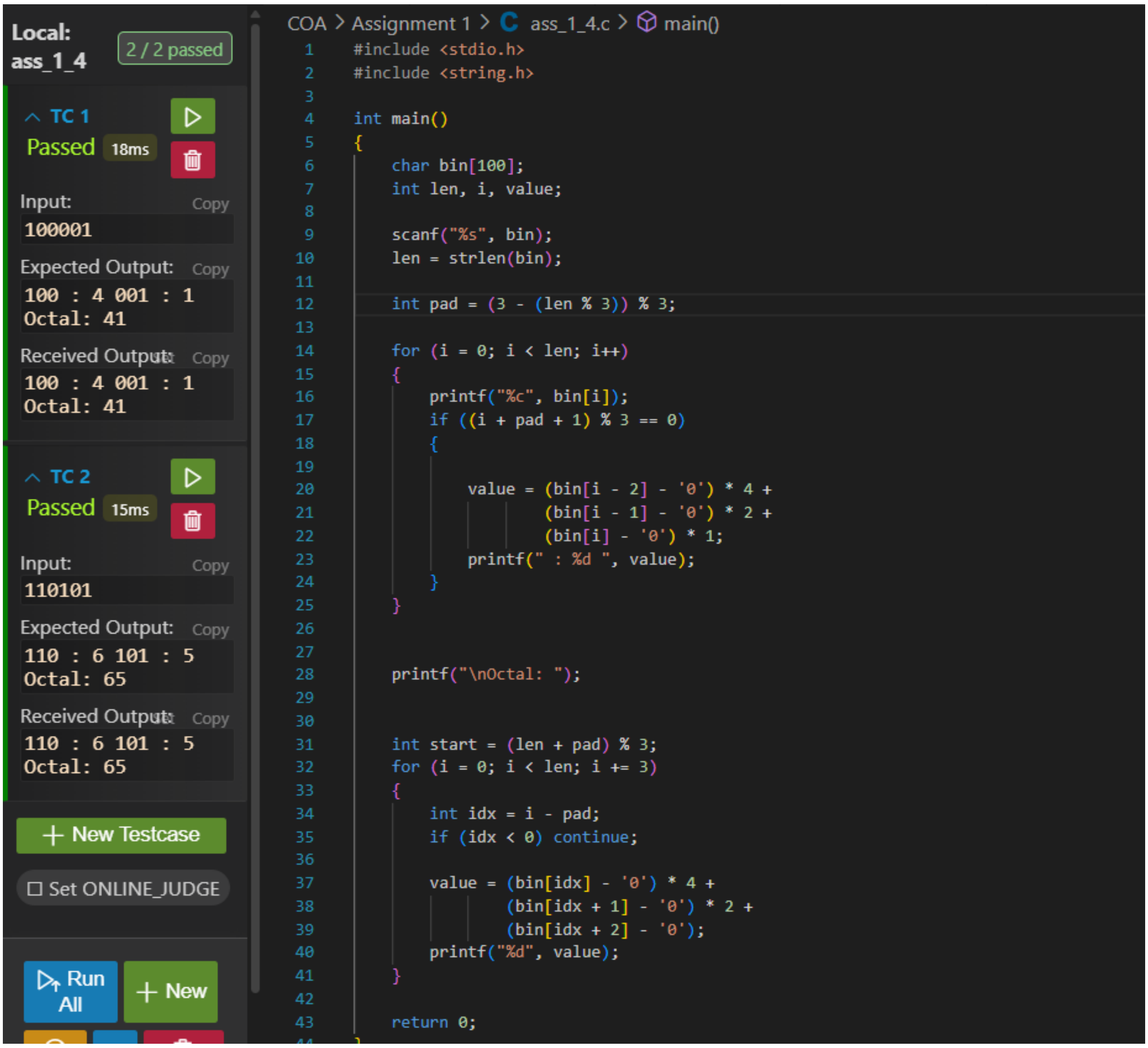
void dec_to_hex(int n)
{
    char hex[100];
    int i = 0;

    if (n == 0)
    {
        printf("0");
        return;
    }

    while (n > 0)
    {
        int rem = n % 16;
        if (rem < 10)
            hex[i++] = rem + '0';
        else
            hex[i++] = rem - 10 + 'A';
        n /= 16;
    }

```c

    for (int j = i - 1; j >= 0; j--)
        printf("%c", hex[j]);
}

int main()
{
    int n;
    scanf("%d", &n);

    printf("Octal: ");
    dec_to_octal(n);

    printf("\nHexadecimal: ");
    dec_to_hex(n);

    return 0;
}
```

Q4



TC 1 Passed 18ms

Input:

```
100001
```

Expected Output:

```
100 : 4 001 : 1
Octal: 41
```

Received Output:

```
100 : 4 001 : 1
Octal: 41
```

TC 2 Passed 15ms

Input:

```
110101
```

Expected Output:

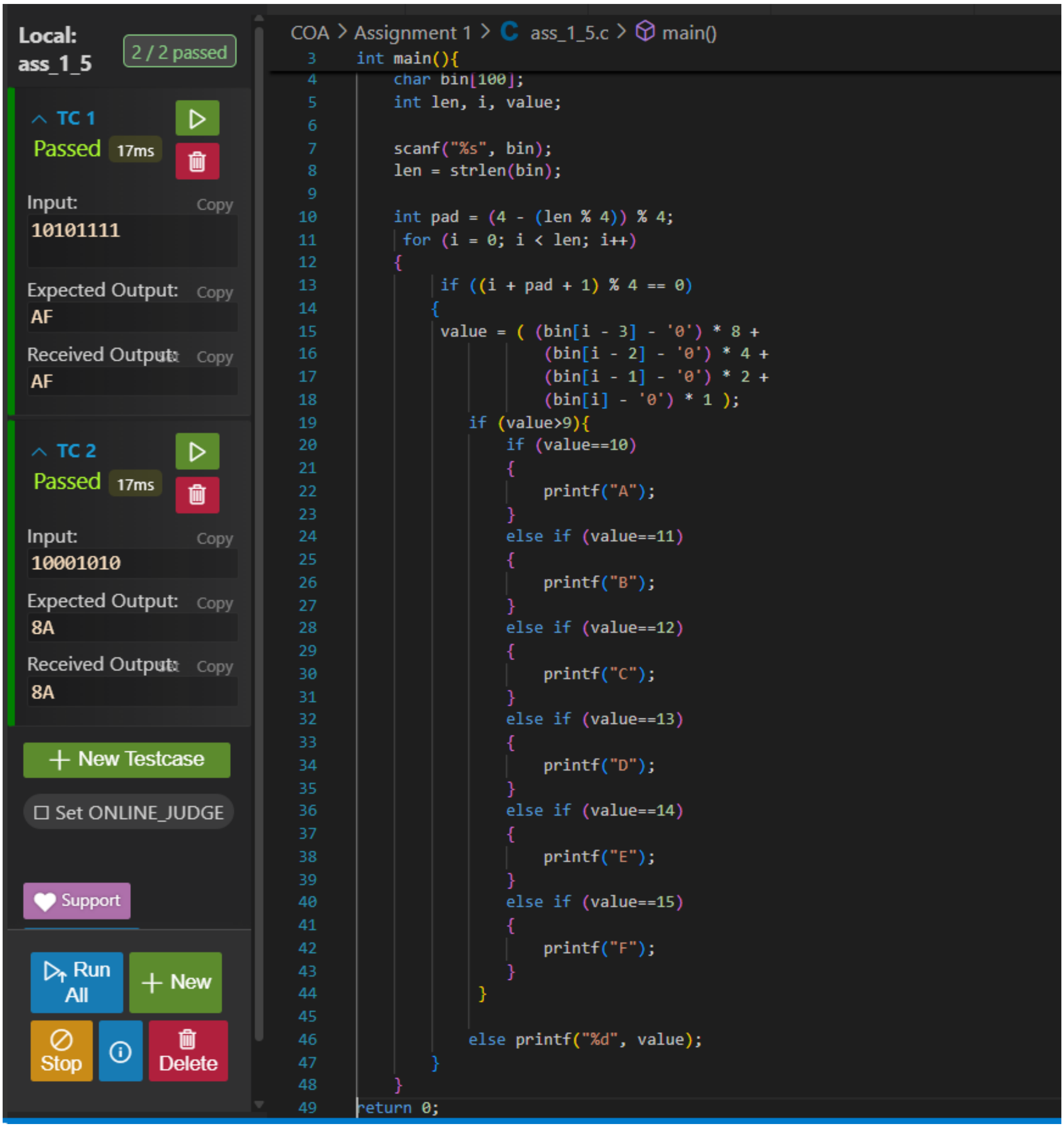
```
110 : 6 101 : 5
Octal: 65
```

Received Output:

```
110 : 6 101 : 5
Octal: 65
```

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char bin[100];
    int len, i, value;

    scanf("%s", bin);
    len = strlen(bin);

    int pad = (3 - (len % 3)) % 3;

    for (i = 0; i < len; i++)
    {
        printf("%c", bin[i]);
        if ((i + pad + 1) % 3 == 0)
        {

            value = (bin[i - 2] - '0') * 4 +
                    (bin[i - 1] - '0') * 2 +
                    (bin[i] - '0') * 1;
            printf(" : %d ", value);
        }
    }


    printf("\nOctal: ");


    int start = (len + pad) % 3;
    for (i = 0; i < len; i += 3)
    {
        int idx = i - pad;
        if (idx < 0) continue;

        value = (bin[idx] - '0') * 4 +
                (bin[idx + 1] - '0') * 2 +
                (bin[idx + 2] - '0');
        printf("%d", value);
    }

    return 0;
}
```

Q5



Local: ass_1_5   2 / 2 passed

TC 1 Passed 17ms

Input: Copy

```
10101111
```

Expected Output: Copy

```
AF
```

Received Output: Copy

```
AF
```

TC 2 Passed 17ms

Input: Copy

```
10001010
```

Expected Output: Copy

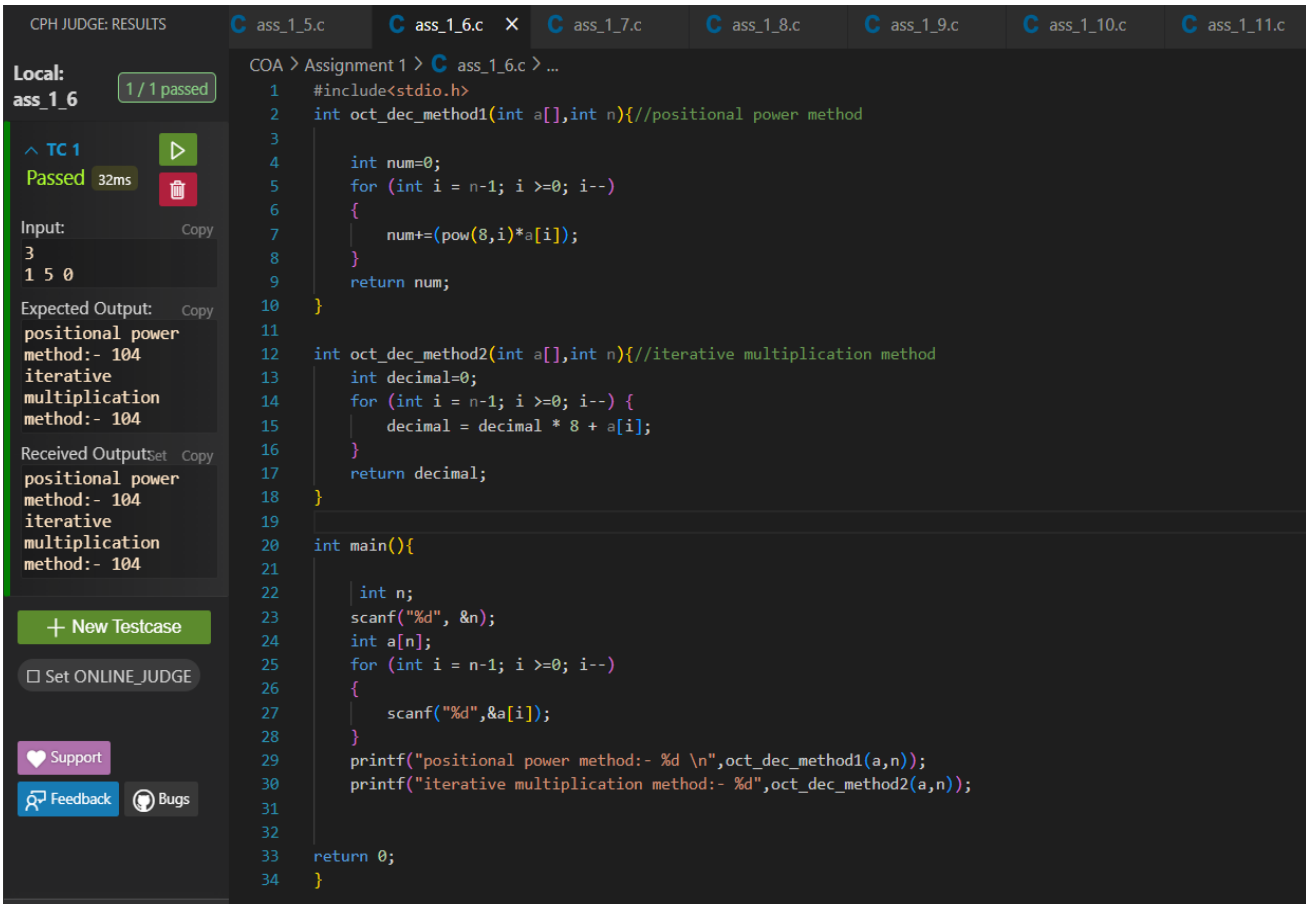
```
8A
```

Received Output: Copy

```
8A
```

```c
int main(){
    char bin[100];
    int len, i, value;

    scanf("%s", bin);
    len = strlen(bin);

    int pad = (4 - (len % 4)) % 4;
     for (i = 0; i < len; i++)
    {
         if ((i + pad + 1) % 4 == 0)
        {
         value = ( (bin[i - 3] - '0') * 8 +
                    (bin[i - 2] - '0') * 4 +
                    (bin[i - 1] - '0') * 2 +
                    (bin[i] - '0') * 1 );
            if (value>9){
                if (value==10)
                {
                    printf("A");
                }
                else if (value==11)
                {
                    printf("B");
                }
                else if (value==12)
                {
                    printf("C");
                }
                else if (value==13)
                {
                    printf("D");
                }
                else if (value==14)
                {
                    printf("E");
                }
                else if (value==15)
                {
                    printf("F");
                }
             }

            else printf("%d", value);
        }
    }
return 0;
```

Q6



Input:

```
3
1 5 0
```

Expected Output:

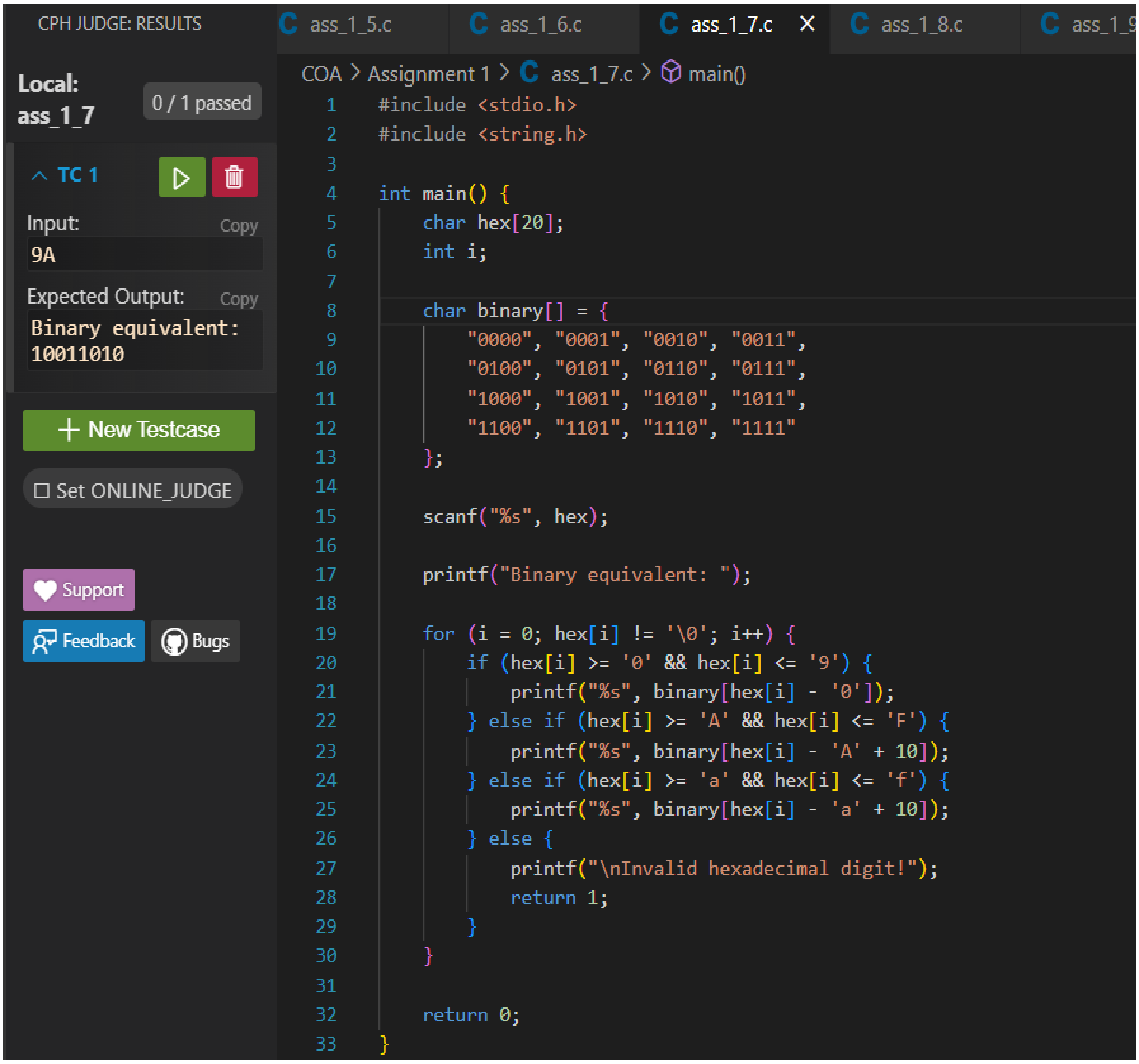
```
positional power
method:- 104
iterative
multiplication
method:- 104
```

Received Output:

```
positional power
method:- 104
iterative
multiplication
method:- 104
```

```c
#include<stdio.h>
int oct_dec_method1(int a[],int n){//positional power method

    int num=0;
    for (int i = n-1; i >=0; i--)
    {
        num+=(pow(8,i)*a[i]);
    }
    return num;
}

int oct_dec_method2(int a[],int n){//iterative multiplication method
    int decimal=0;
    for (int i = n-1; i >=0; i--) {
        decimal = decimal * 8 + a[i];
    }
    return decimal;
}

int main(){

    int n;
    scanf("%d", &n);
    int a[n];
    for (int i = n-1; i >=0; i--)
    {
        scanf("%d",&a[i]);
    }
    printf("positional power method:- %d \n",oct_dec_method1(a,n));
    printf("iterative multiplication method:- %d",oct_dec_method2(a,n));


return 0;
}
```

Q7



Local: ass_1_7

0 / 1 passed

TC 1

Input:

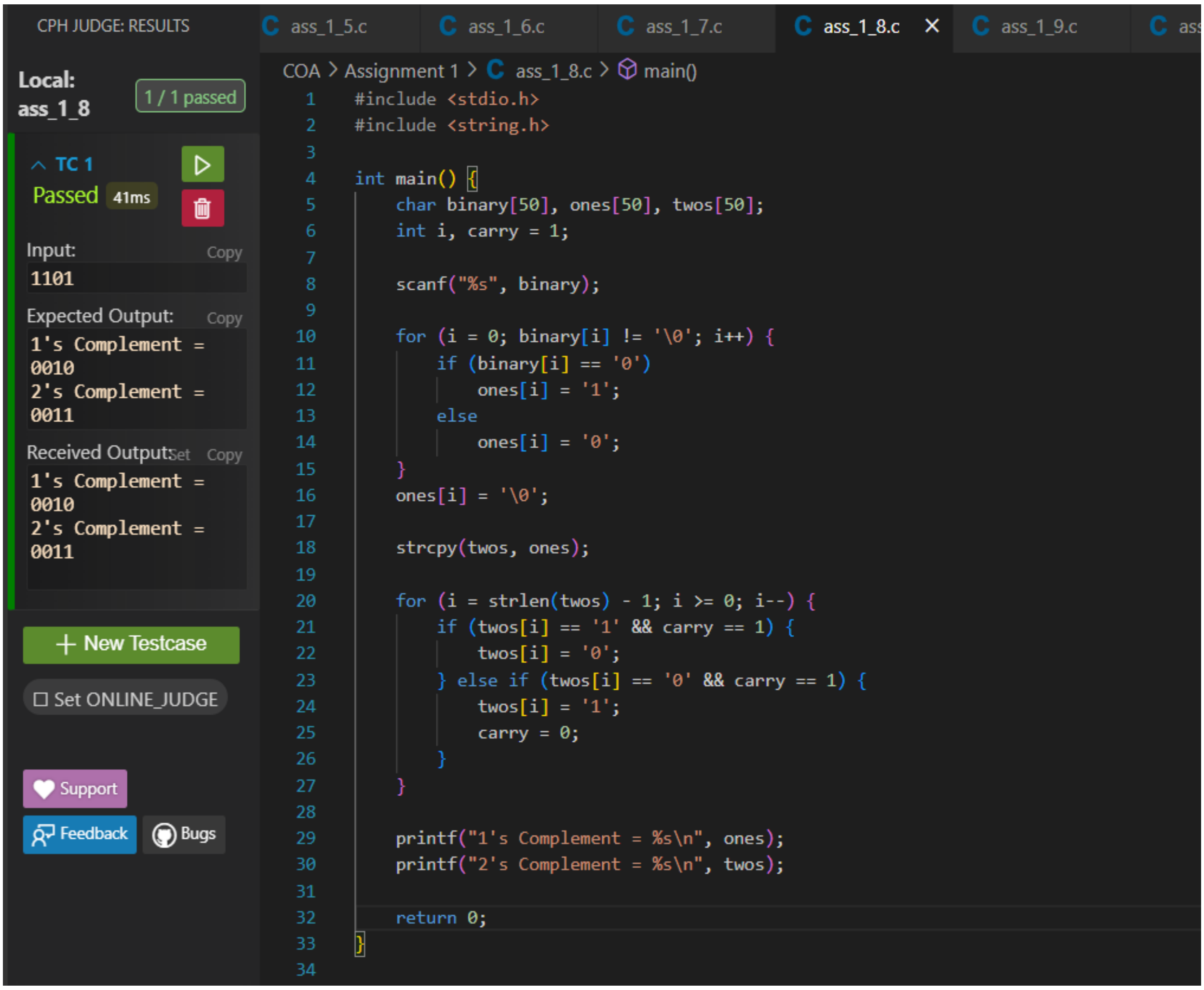
```
9A
```

Expected Output:

```
Binary equivalent:
10011010
```

```c
#include <stdio.h>
#include <string.h>

int main() {
    char hex[20];
    int i;

    char binary[] = {
        "0000", "0001", "0010", "0011",
        "0100", "0101", "0110", "0111",
        "1000", "1001", "1010", "1011",
        "1100", "1101", "1110", "1111"
    };

    scanf("%s", hex);

    printf("Binary equivalent: ");

    for (i = 0; hex[i] != '\0'; i++) {
        if (hex[i] >= '0' && hex[i] <= '9') {
            printf("%s", binary[hex[i] - '0']);
        } else if (hex[i] >= 'A' && hex[i] <= 'F') {
            printf("%s", binary[hex[i] - 'A' + 10]);
        } else if (hex[i] >= 'a' && hex[i] <= 'f') {
            printf("%s", binary[hex[i] - 'a' + 10]);
        } else {
            printf("\nInvalid hexadecimal digit!");
            return 1;
        }
    }

    return 0;
}
```

Q8



Input:

```
1101
```

Expected Output:

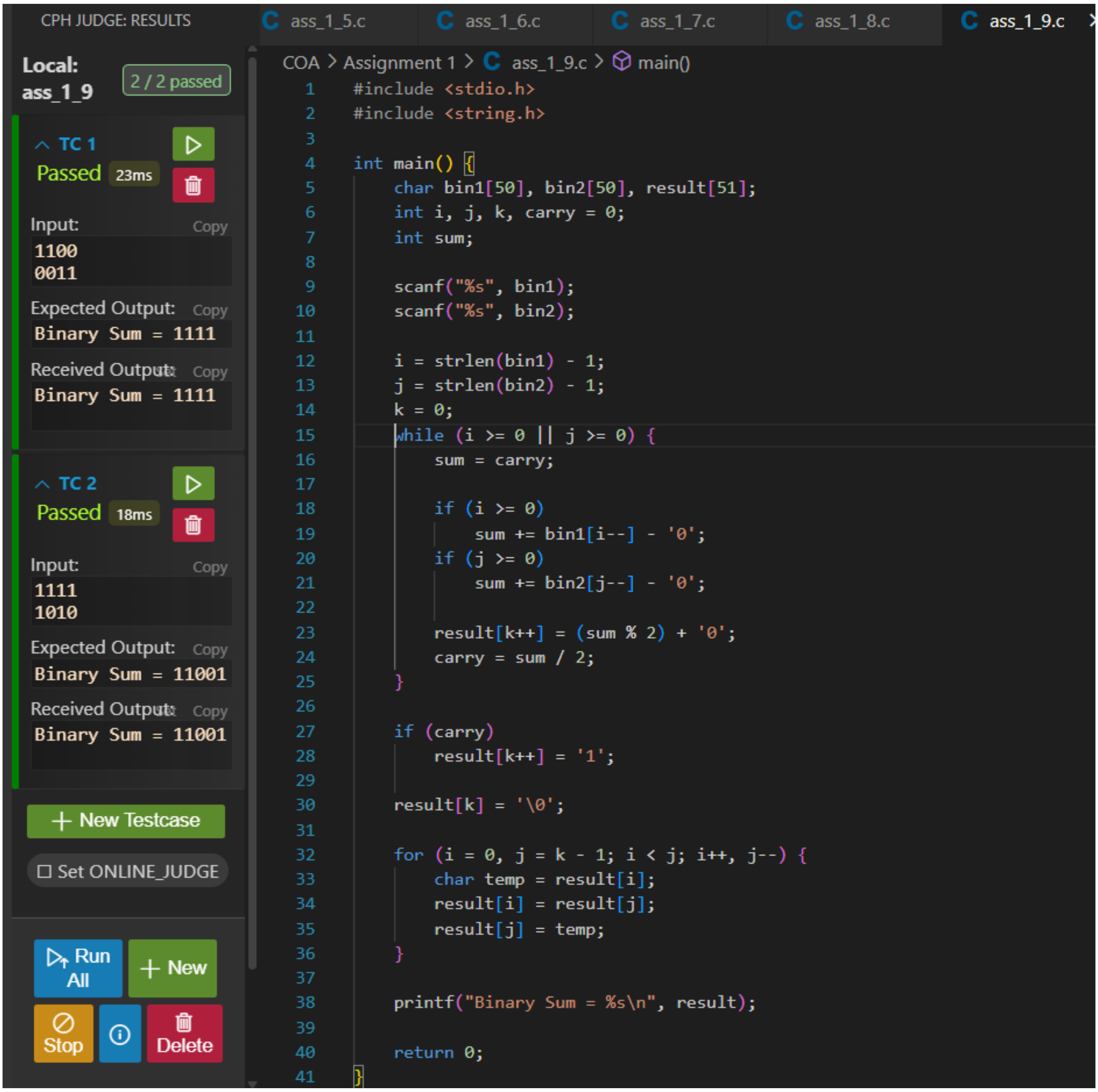
```
1's Complement = 0010
2's Complement = 0011
```

Received Output:

```
1's Complement = 0010
2's Complement = 0011
```

```c
#include <stdio.h>
#include <string.h>

int main() {
    char binary[50], ones[50], twos[50];
    int i, carry = 1;

    scanf("%s", binary);

    for (i = 0; binary[i] != '\0'; i++) {
        if (binary[i] == '0')
            ones[i] = '1';
        else
            ones[i] = '0';
    }
    ones[i] = '\0';

    strcpy(twos, ones);

    for (i = strlen(twos) - 1; i >= 0; i--) {
        if (twos[i] == '1' && carry == 1) {
            twos[i] = '0';
        } else if (twos[i] == '0' && carry == 1) {
            twos[i] = '1';
            carry = 0;
        }
    }

    printf("1's Complement = %s\n", ones);
    printf("2's Complement = %s\n", twos);

    return 0;
}
```

Q9



TC 1 Passed 23ms

Input:

```
1100
0011
```

Expected Output:

```
Binary Sum = 1111
```

Received Output:

```
Binary Sum = 1111
```

TC 2 Passed 18ms

Input:

```
1111
1010
```

Expected Output:

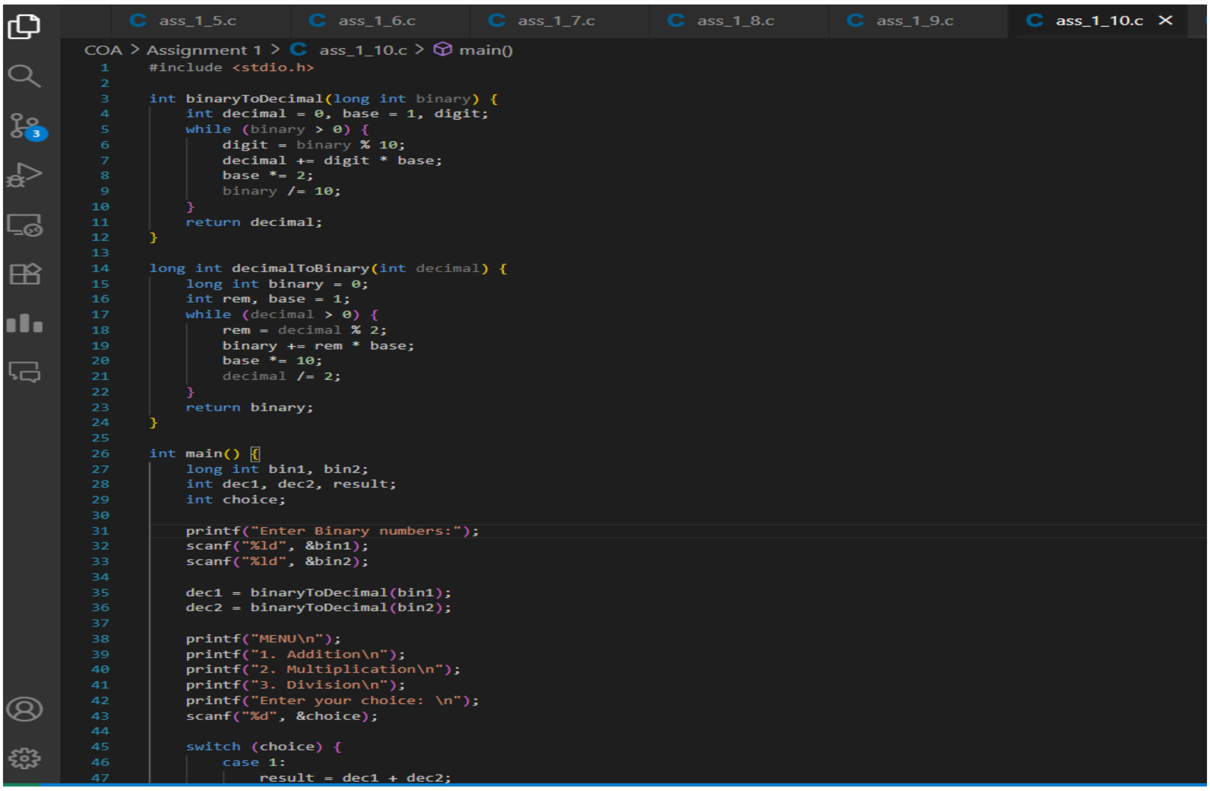
```
Binary Sum = 11001
```

Received Output:

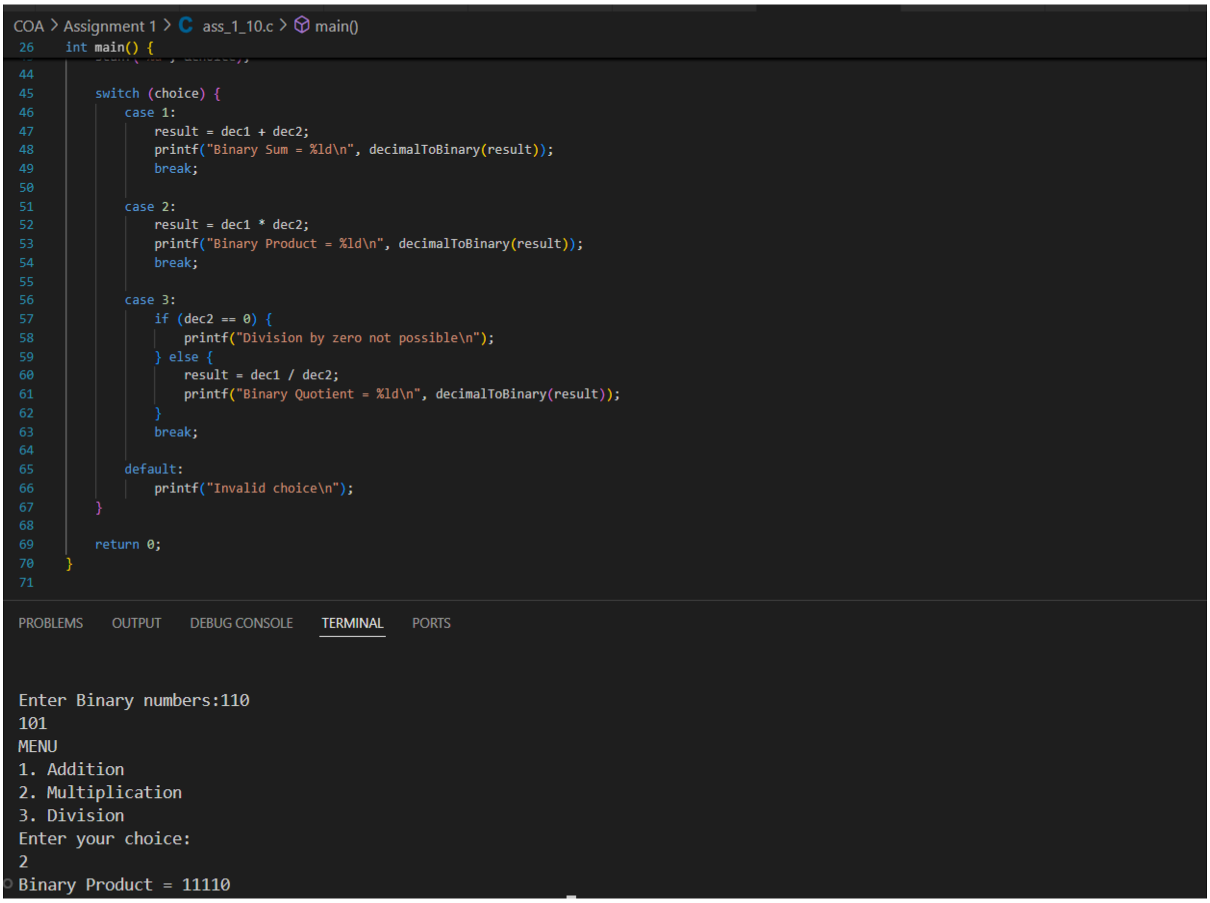
```
Binary Sum = 11001
```

```c
#include <stdio.h>
#include <string.h>

int main() {
    char bin1[50], bin2[50], result[51];
    int i, j, k, carry = 0;
    int sum;

    scanf("%s", bin1);
    scanf("%s", bin2);

    i = strlen(bin1) - 1;
    j = strlen(bin2) - 1;
    k = 0;
    while (i >= 0 || j >= 0) {
        sum = carry;

        if (i >= 0)
            sum += bin1[i--] - '0';
        if (j >= 0)
            sum += bin2[j--] - '0';

        result[k++] = (sum % 2) + '0';
        carry = sum / 2;
    }

    if (carry)
        result[k++] = '1';

    result[k] = '\0';

    for (i = 0, j = k - 1; i < j; i++, j--) {
        char temp = result[i];
        result[i] = result[j];
        result[j] = temp;
    }

    printf("Binary Sum = %s\n", result);

    return 0;
}
```

Q10



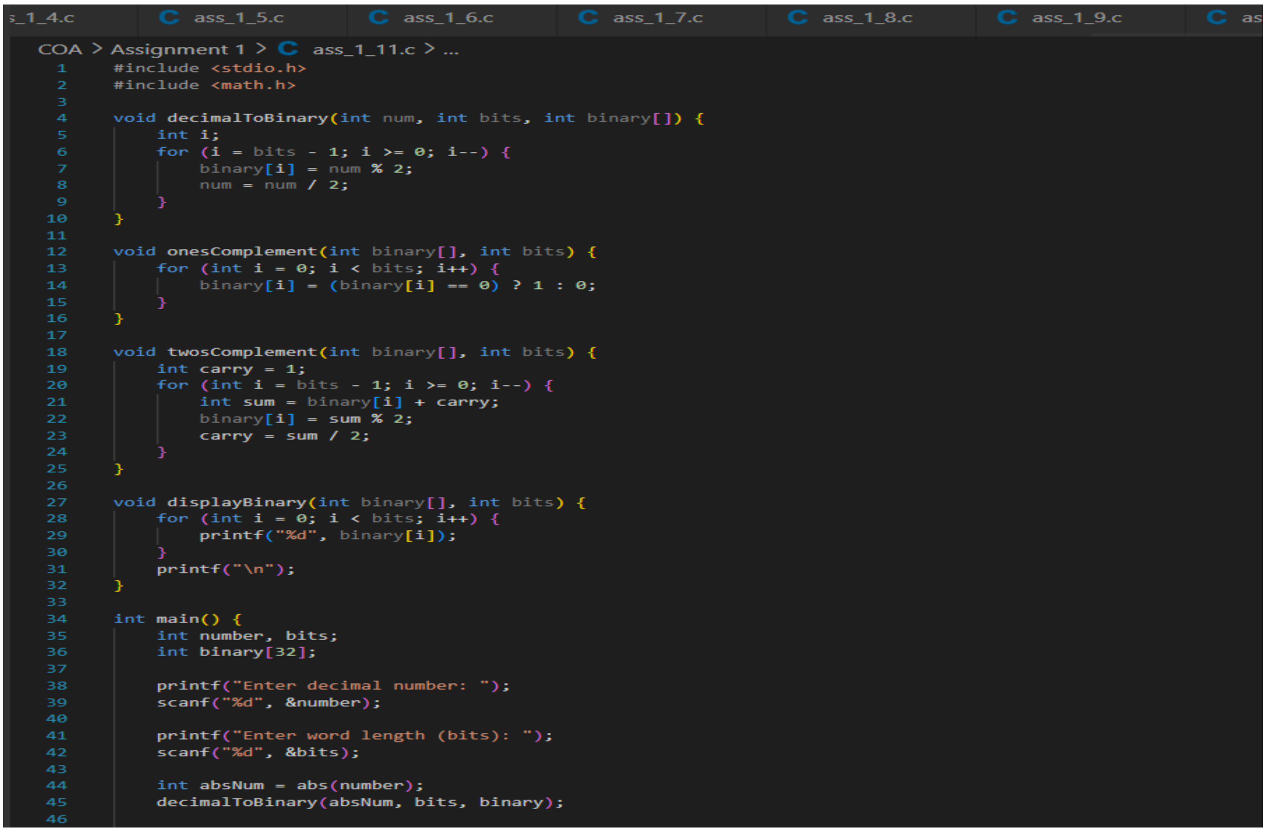
```c
#include <stdio.h>

int binaryToDecimal(long int binary) {
    int decimal = 0, base = 1, digit;
    while (binary > 0) {
        digit = binary % 10;
        decimal += digit * base;
        base *= 2;
        binary /= 10;
    }
    return decimal;
}

long int decimalToBinary(int decimal) {
    long int binary = 0;
    int rem, base = 1;
    while (decimal > 0) {
        rem = decimal % 2;
        binary += rem * base;
        base *= 10;
        decimal /= 2;
    }
    return binary;
}

int main() {
    long int bin1, bin2;
    int dec1, dec2, result;
    int choice;

    printf("Enter Binary numbers:");
    scanf("%ld", &bin1);
    scanf("%ld", &bin2);

    dec1 = binaryToDecimal(bin1);
    dec2 = binaryToDecimal(bin2);

    printf("MENU\n");
    printf("1. Addition\n");
    printf("2. Multiplication\n");
    printf("3. Division\n");
    printf("Enter your choice: \n");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            result = dec1 + dec2;
```



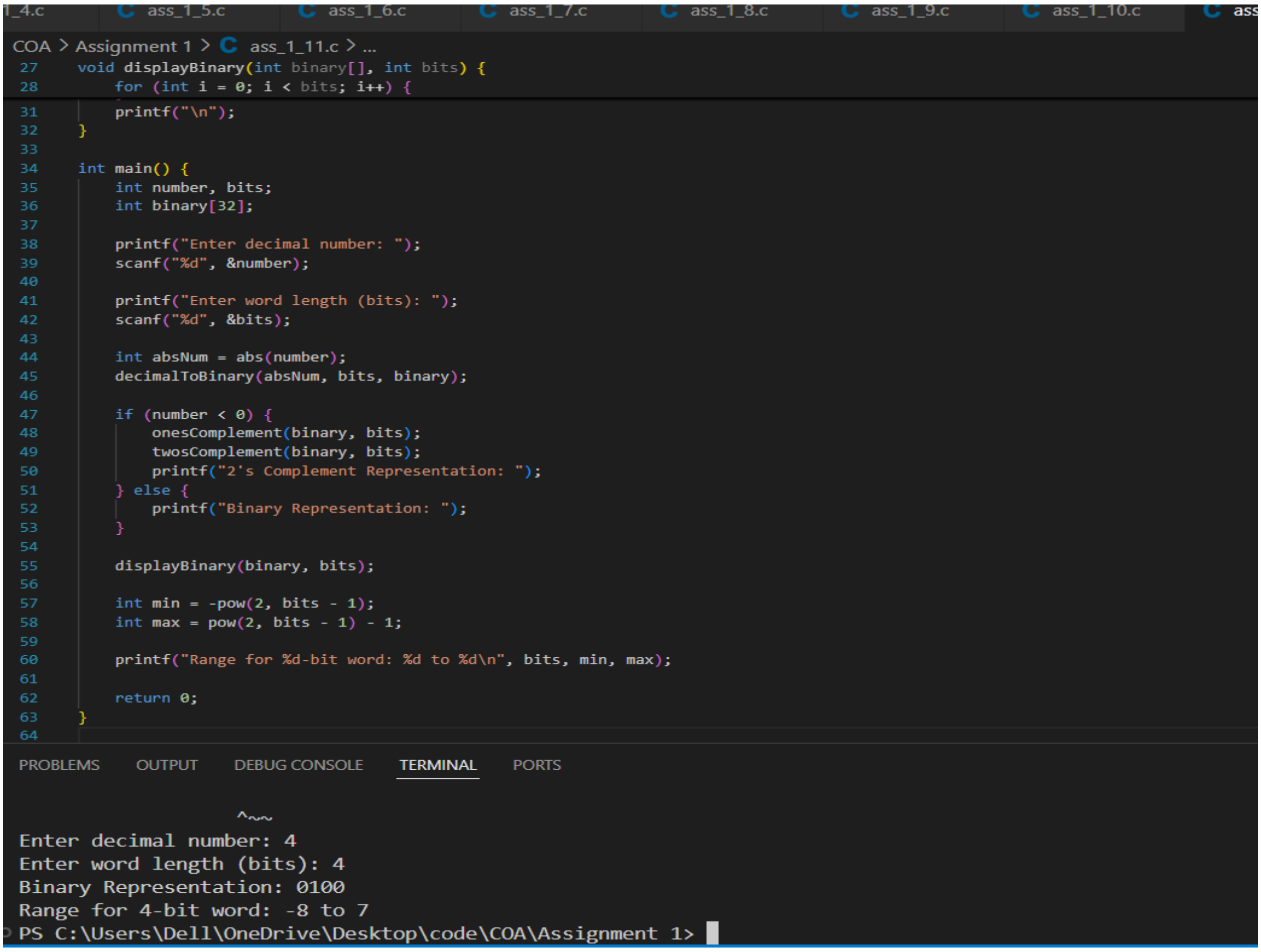
```c
int main() {

    switch (choice) {
        case 1:
            result = dec1 + dec2;
            printf("Binary Sum = %ld\n", decimalToBinary(result));
            break;

        case 2:
            result = dec1 * dec2;
            printf("Binary Product = %ld\n", decimalToBinary(result));
            break;

        case 3:
            if (dec2 == 0) {
                printf("Division by zero not possible\n");
            } else {
                result = dec1 / dec2;
                printf("Binary Quotient = %ld\n", decimalToBinary(result));
            }
            break;

        default:
            printf("Invalid choice\n");
    }

    return 0;
}
```

```
Enter Binary numbers:110
101
MENU
1. Addition
2. Multiplication
3. Division
Enter your choice:
2
Binary Product = 11110
```

Q11



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

void decimalToBinary(int num, int bits, int binary[]) {
    int i;
    for (i = bits - 1; i >= 0; i--) {
        binary[i] = num % 2;
        num = num / 2;
    }
}

void onesComplement(int binary[], int bits) {
    for (int i = 0; i < bits; i++) {
        binary[i] = (binary[i] == 0) ? 1 : 0;
    }
}

void twosComplement(int binary[], int bits) {
    int carry = 1;
    for (int i = bits - 1; i >= 0; i--) {
        int sum = binary[i] + carry;
        binary[i] = sum % 2;
        carry = sum / 2;
    }
}

void displayBinary(int binary[], int bits) {
    for (int i = 0; i < bits; i++) {
        printf("%d", binary[i]);
    }
    printf("\n");
}

int main() {
    int number, bits;
    int binary[32];

    printf("Enter decimal number: ");
    scanf("%d", &number);

    printf("Enter word length (bits): ");
    scanf("%d", &bits);

    int absNum = abs(number);
    decimalToBinary(absNum, bits, binary);

```

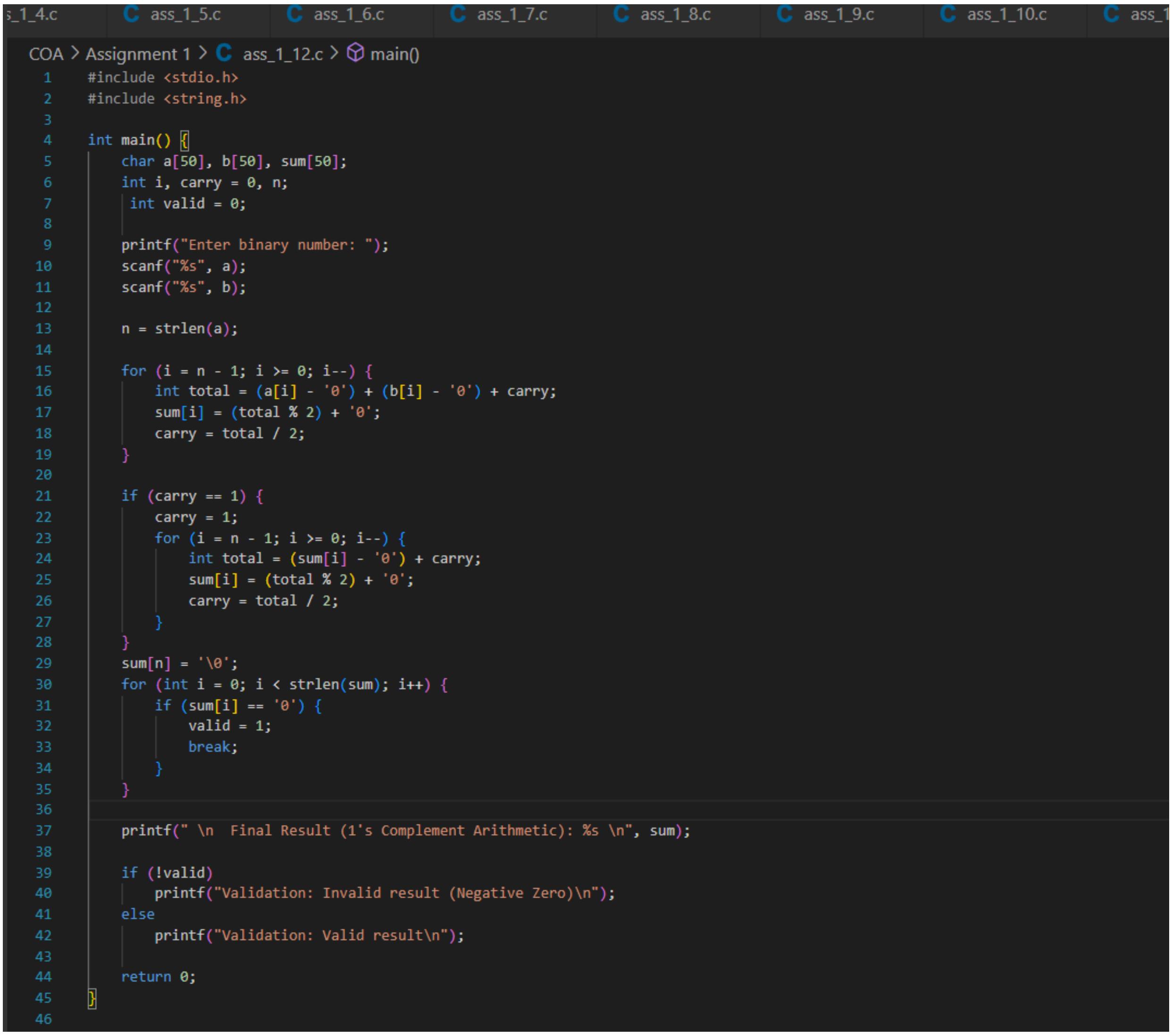


```c
void displayBinary(int binary[], int bits) {
    for (int i = 0; i < bits; i++) {
    printf("\n");
}

int main() {
    int number, bits;
    int binary[32];

    printf("Enter decimal number: ");
    scanf("%d", &number);

    printf("Enter word length (bits): ");
    scanf("%d", &bits);

    int absNum = abs(number);
    decimalToBinary(absNum, bits, binary);

```

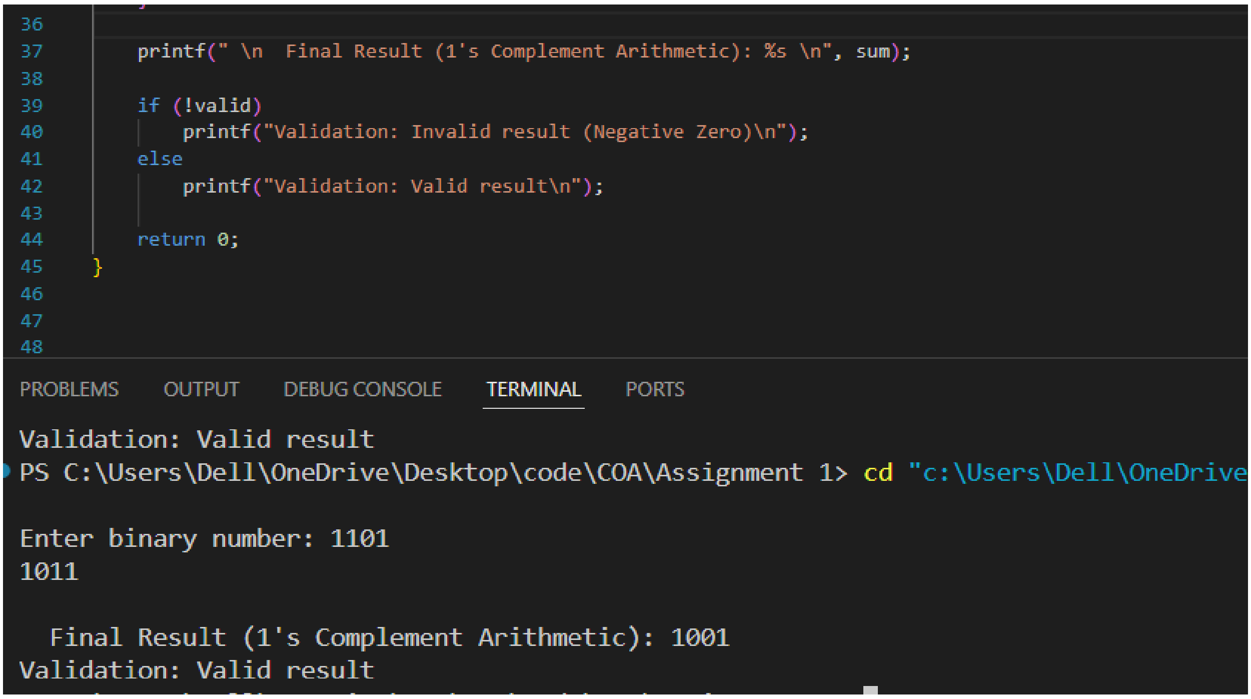
```c
    if (number < 0) {
        onesComplement(binary, bits);
        twosComplement(binary, bits);
        printf("2's Complement Representation: ");
    } else {
        printf("Binary Representation: ");
    }

    displayBinary(binary, bits);

    int min = -pow(2, bits - 1);
    int max = pow(2, bits - 1) - 1;

    printf("Range for %d-bit word: %d to %d\n", bits, min, max);

    return 0;
}

```

```
                      ^~~
Enter decimal number: 4
Enter word length (bits): 4
Binary Representation: 0100
Range for 4-bit word: -8 to 7
PS C:\Users\Dell\OneDrive\Desktop\code\COA\Assignment 1>
```

Q12



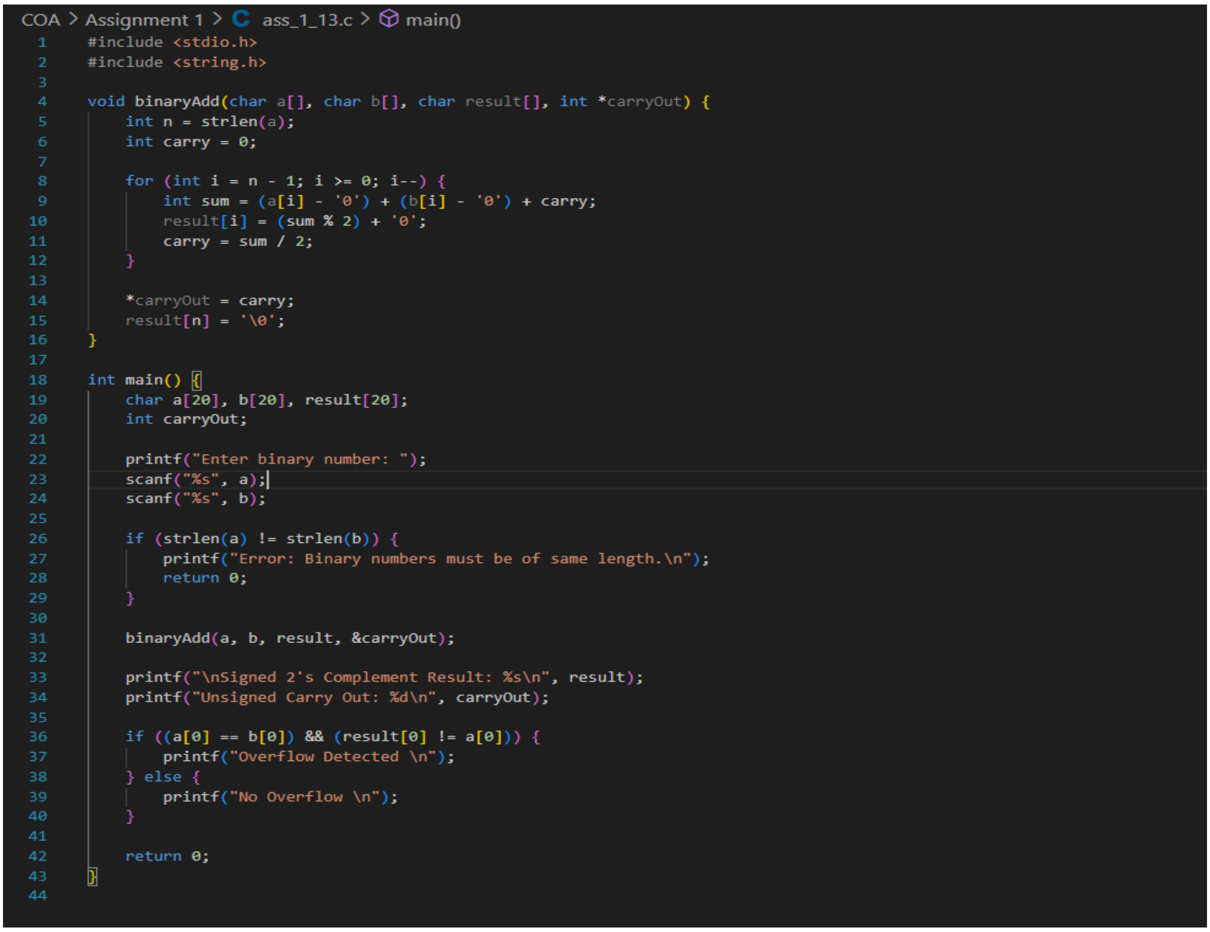
```c
#include <stdio.h>
#include <string.h>

int main() {
    char a[50], b[50], sum[50];
    int i, carry = 0, n;
     int valid = 0;

    printf("Enter binary number: ");
    scanf("%s", a);
    scanf("%s", b);

    n = strlen(a);

    for (i = n - 1; i >= 0; i--) {
        int total = (a[i] - '0') + (b[i] - '0') + carry;
        sum[i] = (total % 2) + '0';
        carry = total / 2;
    }

    if (carry == 1) {
        carry = 1;
        for (i = n - 1; i >= 0; i--) {
            int total = (sum[i] - '0') + carry;
            sum[i] = (total % 2) + '0';
            carry = total / 2;
        }
    }
    sum[n] = '\0';
    for (int i = 0; i < strlen(sum); i++) {
        if (sum[i] == '0') {
            valid = 1;
            break;
        }
    }

    printf(" \n  Final Result (1's Complement Arithmetic): %s \n", sum);

    if (!valid)
        printf("Validation: Invalid result (Negative Zero)\n");
    else
        printf("Validation: Valid result\n");

    return 0;
}
```



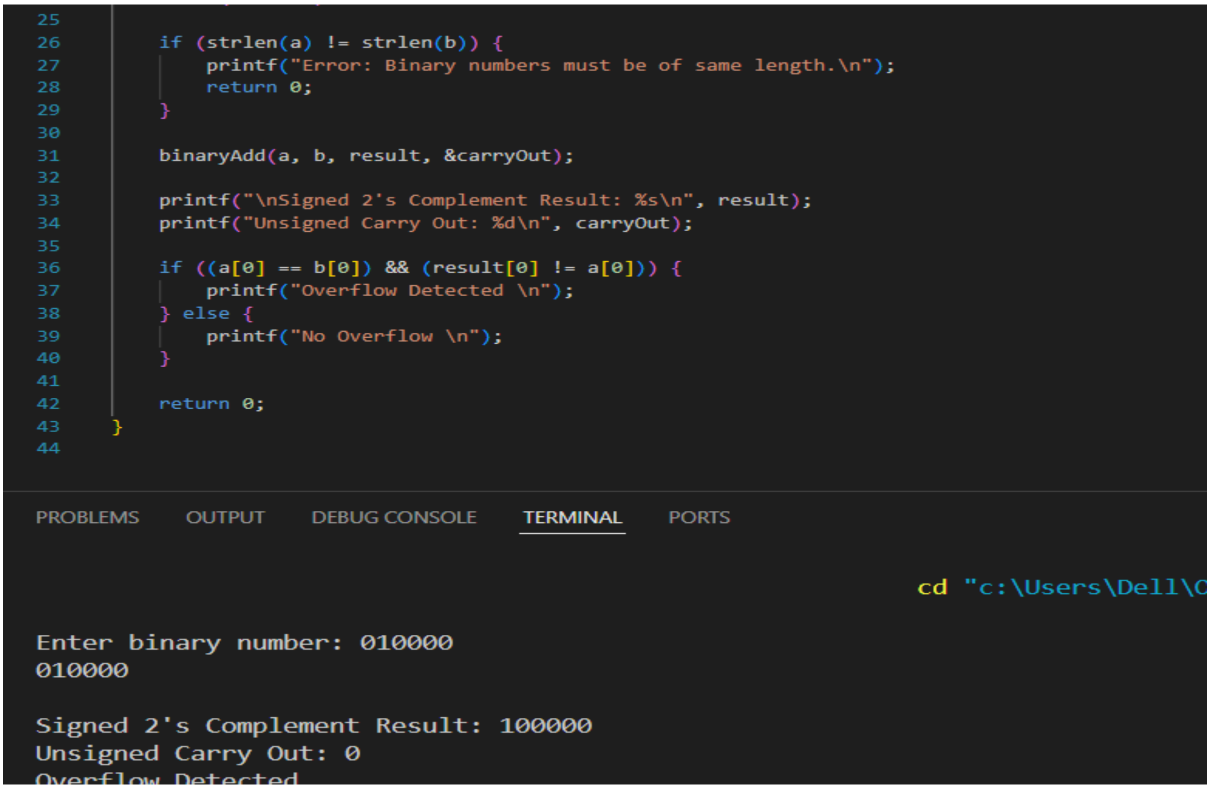
```c

    printf(" \n  Final Result (1's Complement Arithmetic): %s \n", sum);

    if (!valid)
        printf("Validation: Invalid result (Negative Zero)\n");
    else
        printf("Validation: Valid result\n");

    return 0;
}
```

```
Validation: Valid result
PS C:\Users\Dell\OneDrive\Desktop\code\COA\Assignment 1> cd "c:\Users\Dell\OneDrive

Enter binary number: 1101
1011

  Final Result (1's Complement Arithmetic): 1001
Validation: Valid result
```

Q13



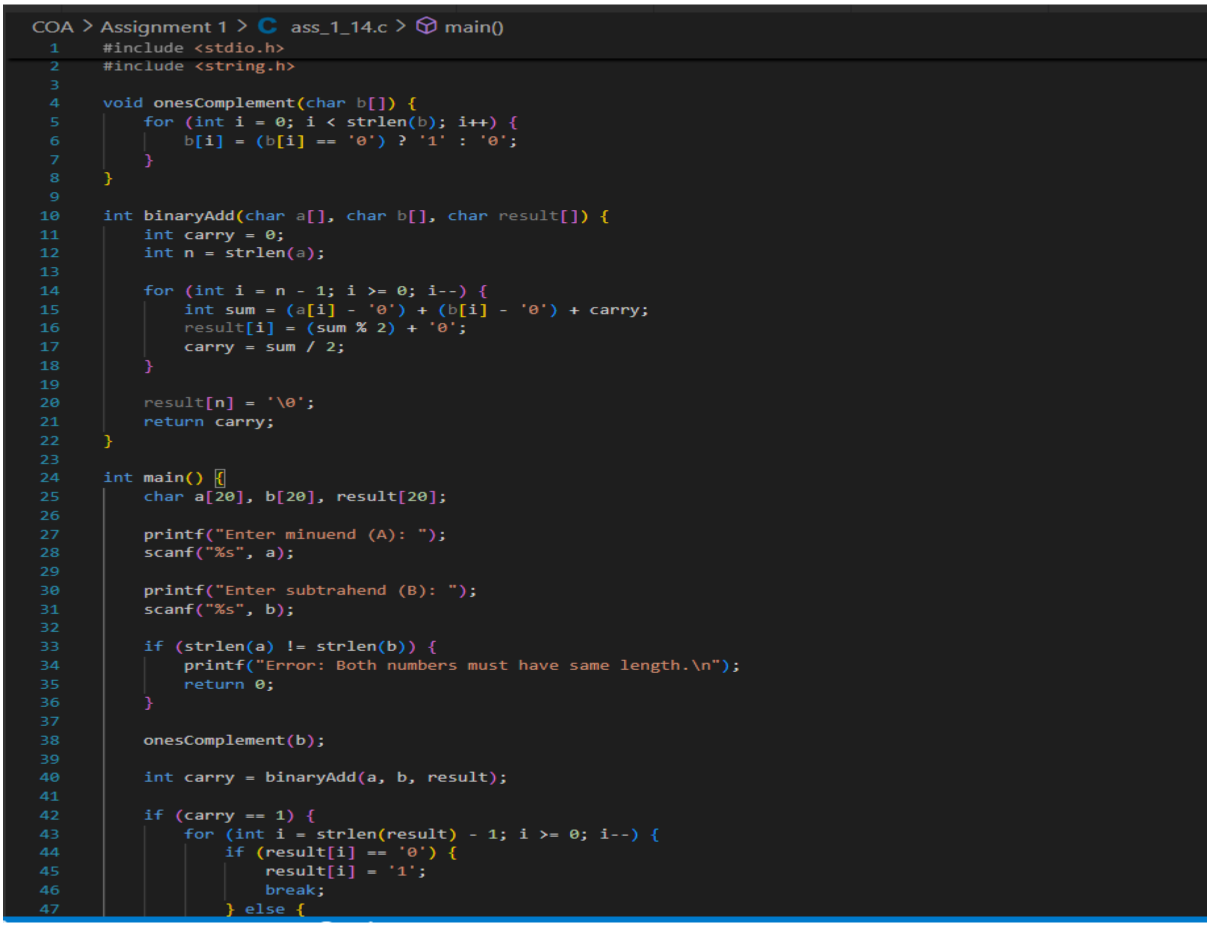
```c
#include <stdio.h>
#include <string.h>

void binaryAdd(char a[], char b[], char result[], int *carryOut) {
    int n = strlen(a);
    int carry = 0;

    for (int i = n - 1; i >= 0; i--) {
        int sum = (a[i] - '0') + (b[i] - '0') + carry;
        result[i] = (sum % 2) + '0';
        carry = sum / 2;
    }

    *carryOut = carry;
    result[n] = '\0';
}

int main() {
    char a[20], b[20], result[20];
    int carryOut;

    printf("Enter binary number: ");
    scanf("%s", a);
    scanf("%s", b);

    if (strlen(a) != strlen(b)) {
        printf("Error: Binary numbers must be of same length.\n");
        return 0;
    }

    binaryAdd(a, b, result, &carryOut);

    printf("\nSigned 2's Complement Result: %s\n", result);
    printf("Unsigned Carry Out: %d\n", carryOut);

    if ((a[0] == b[0]) && (result[0] != a[0])) {
        printf("Overflow Detected \n");
    } else {
        printf("No Overflow \n");
    }

    return 0;
}
```



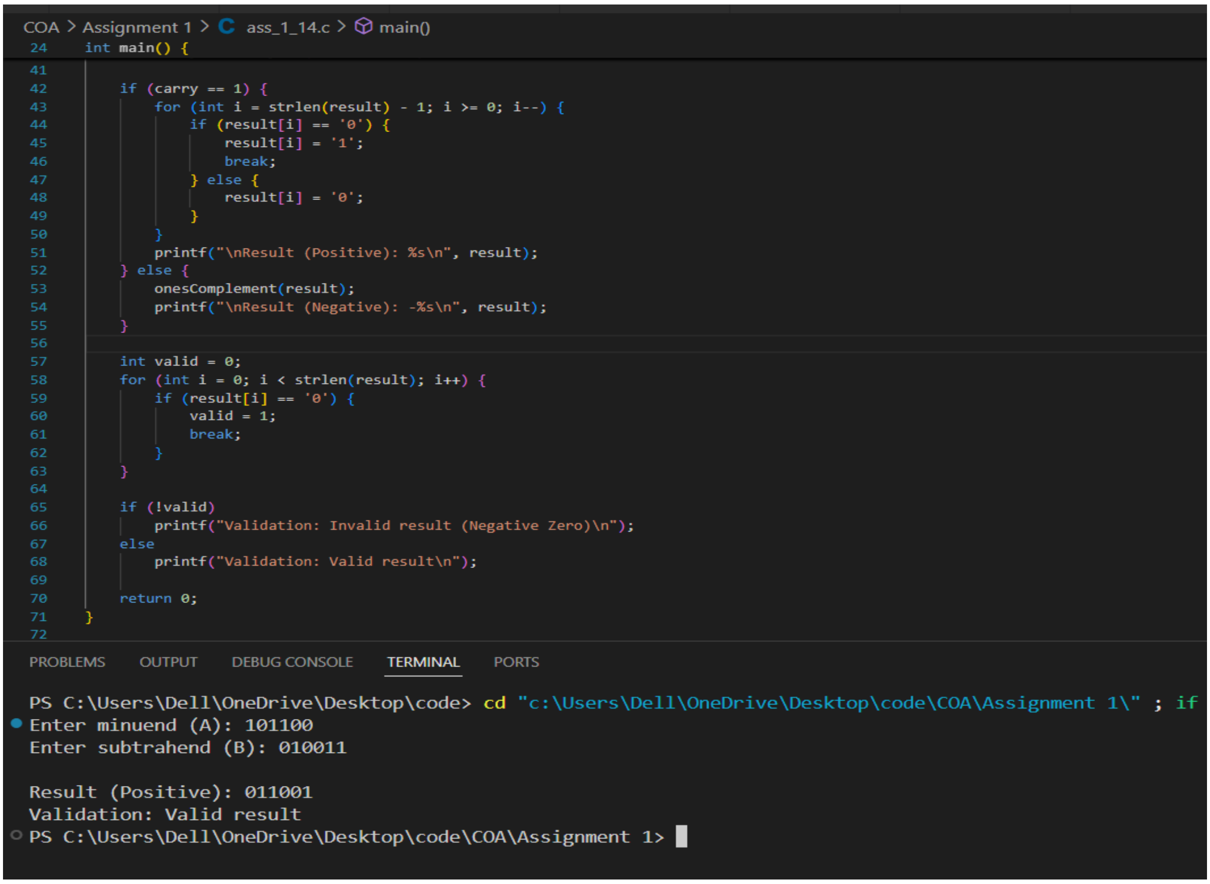
```c
    if (strlen(a) != strlen(b)) {
        printf("Error: Binary numbers must be of same length.\n");
        return 0;
    }

    binaryAdd(a, b, result, &carryOut);

    printf("\nSigned 2's Complement Result: %s\n", result);
    printf("Unsigned Carry Out: %d\n", carryOut);

    if ((a[0] == b[0]) && (result[0] != a[0])) {
        printf("Overflow Detected \n");
    } else {
        printf("No Overflow \n");
    }

    return 0;
}
```

```
                                                          cd "c:\Users\Dell\C

Enter binary number: 010000
010000

Signed 2's Complement Result: 100000
Unsigned Carry Out: 0
Overflow Detected
```

Q14



```c
#include <stdio.h>
#include <string.h>

void onesComplement(char b[]) {
    for (int i = 0; i < strlen(b); i++) {
        b[i] = (b[i] == '0') ? '1' : '0';
    }
}

int binaryAdd(char a[], char b[], char result[]) {
    int carry = 0;
    int n = strlen(a);

    for (int i = n - 1; i >= 0; i--) {
        int sum = (a[i] - '0') + (b[i] - '0') + carry;
        result[i] = (sum % 2) + '0';
        carry = sum / 2;
    }

    result[n] = '\0';
    return carry;
}

int main() {
    char a[20], b[20], result[20];

    printf("Enter minuend (A): ");
    scanf("%s", a);

    printf("Enter subtrahend (B): ");
    scanf("%s", b);

    if (strlen(a) != strlen(b)) {
        printf("Error: Both numbers must have same length.\n");
        return 0;
    }

    onesComplement(b);

    int carry = binaryAdd(a, b, result);

    if (carry == 1) {
        for (int i = strlen(result) - 1; i >= 0; i--) {
            if (result[i] == '0') {
                result[i] = '1';
                break;
            } else {
```



```c
int main() {

    if (carry == 1) {
        for (int i = strlen(result) - 1; i >= 0; i--) {
            if (result[i] == '0') {
                result[i] = '1';
                break;
            } else {
                result[i] = '0';
            }
        }
        printf("\nResult (Positive): %s\n", result);
    } else {
        onesComplement(result);
        printf("\nResult (Negative): -%s\n", result);
    }

    int valid = 0;
    for (int i = 0; i < strlen(result); i++) {
        if (result[i] == '0') {
            valid = 1;
            break;
        }
    }

    if (!valid)
        printf("Validation: Invalid result (Negative Zero)\n");
    else
        printf("Validation: Valid result\n");

    return 0;
}
```

```
PS C:\Users\Dell\OneDrive\Desktop\code> cd "c:\Users\Dell\OneDrive\Desktop\code\COA\Assignment 1\" ; if
Enter minuend (A): 101100
Enter subtrahend (B): 010011

Result (Positive): 011001
Validation: Valid result
PS C:\Users\Dell\OneDrive\Desktop\code\COA\Assignment 1>
```