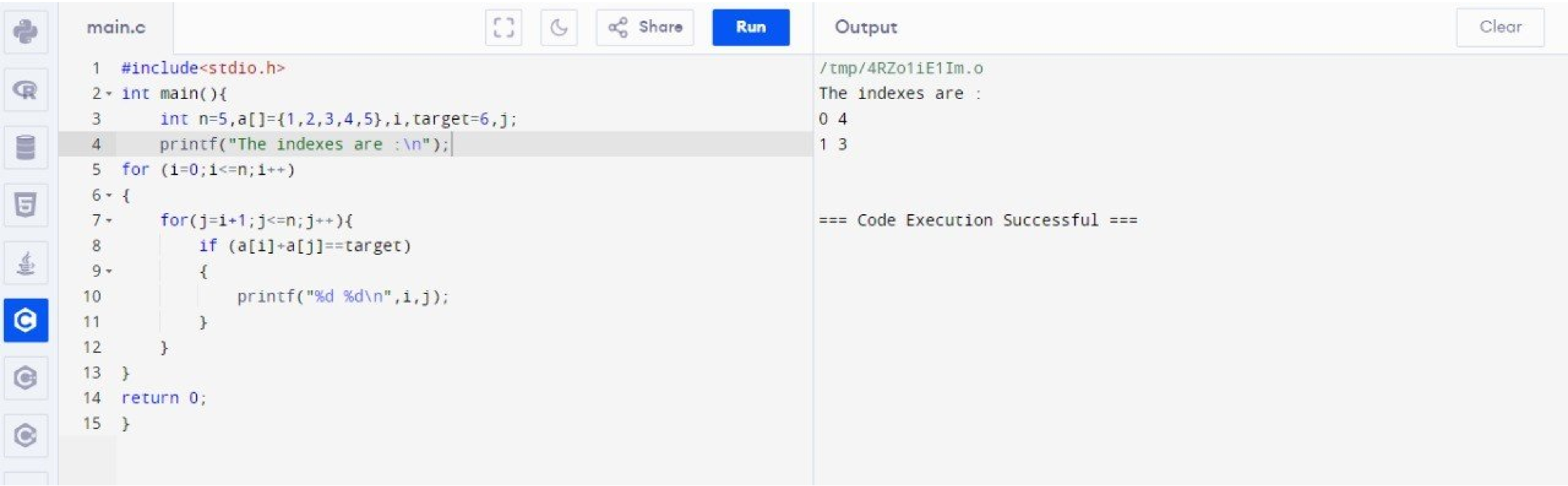
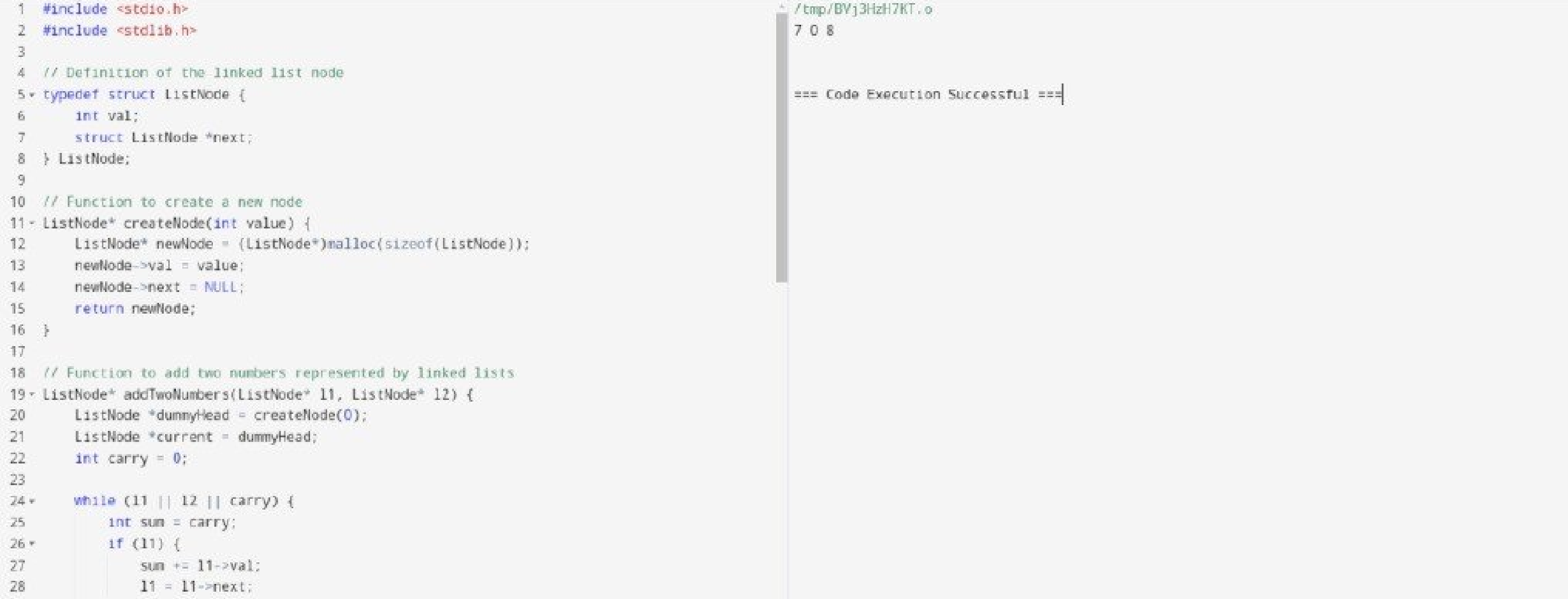


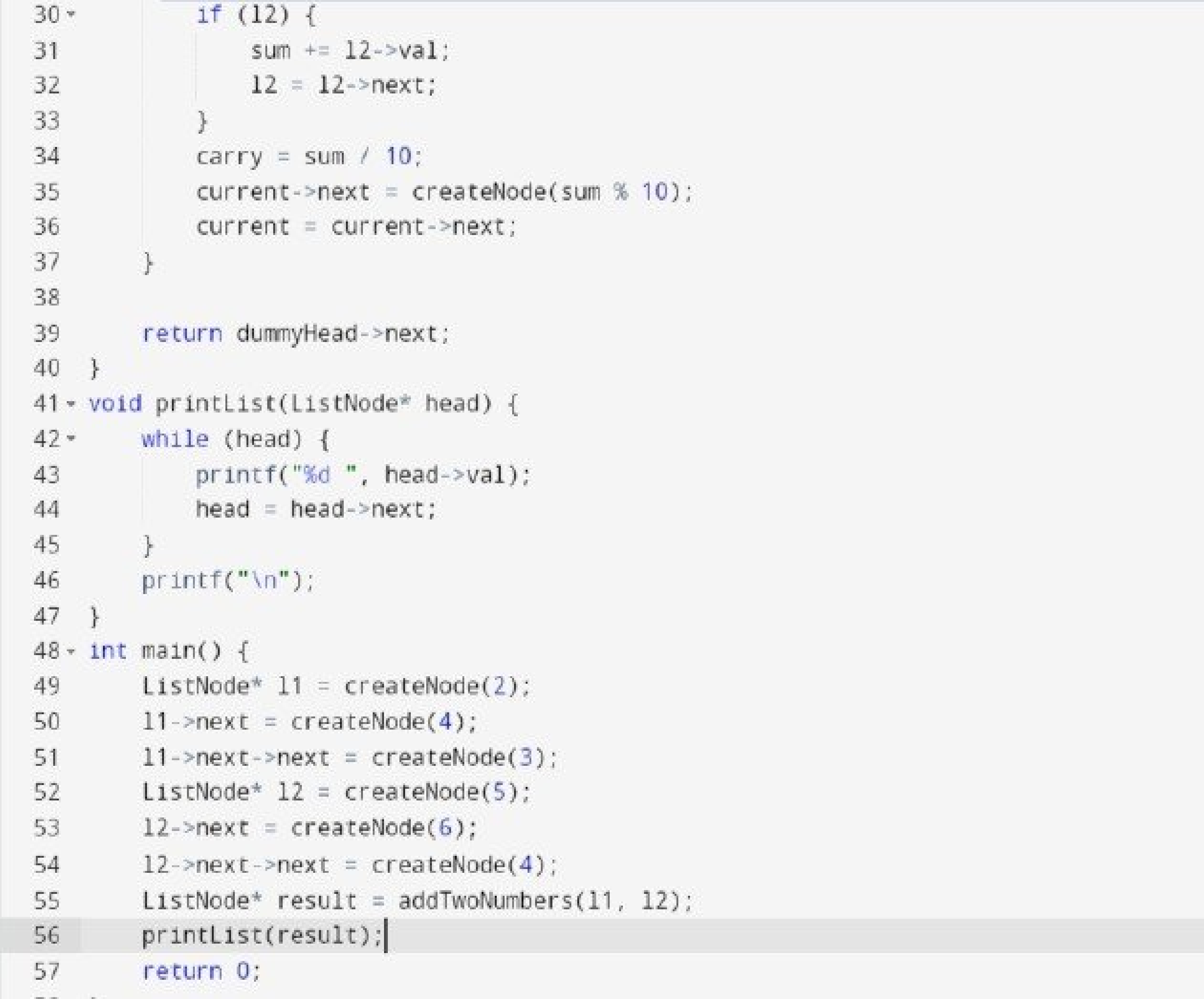
```c
#include<stdio.h>
int main(){
    int n=5,a[]={1,2,3,4,5},i,target=6,j;
    printf("The indexes are :\n");
for (i=0;i<=n;i++)
{
    for(j=i+1;j<=n;j++){
        if (a[i]+a[j]==target)
        {
            printf("%d %d\n",i,j);
        }
    }
}
return 0;
}
```

```
/tmp/4RZo1iE1Im.o
The indexes are :
0 4
1 3


=== Code Execution Successful ===
```

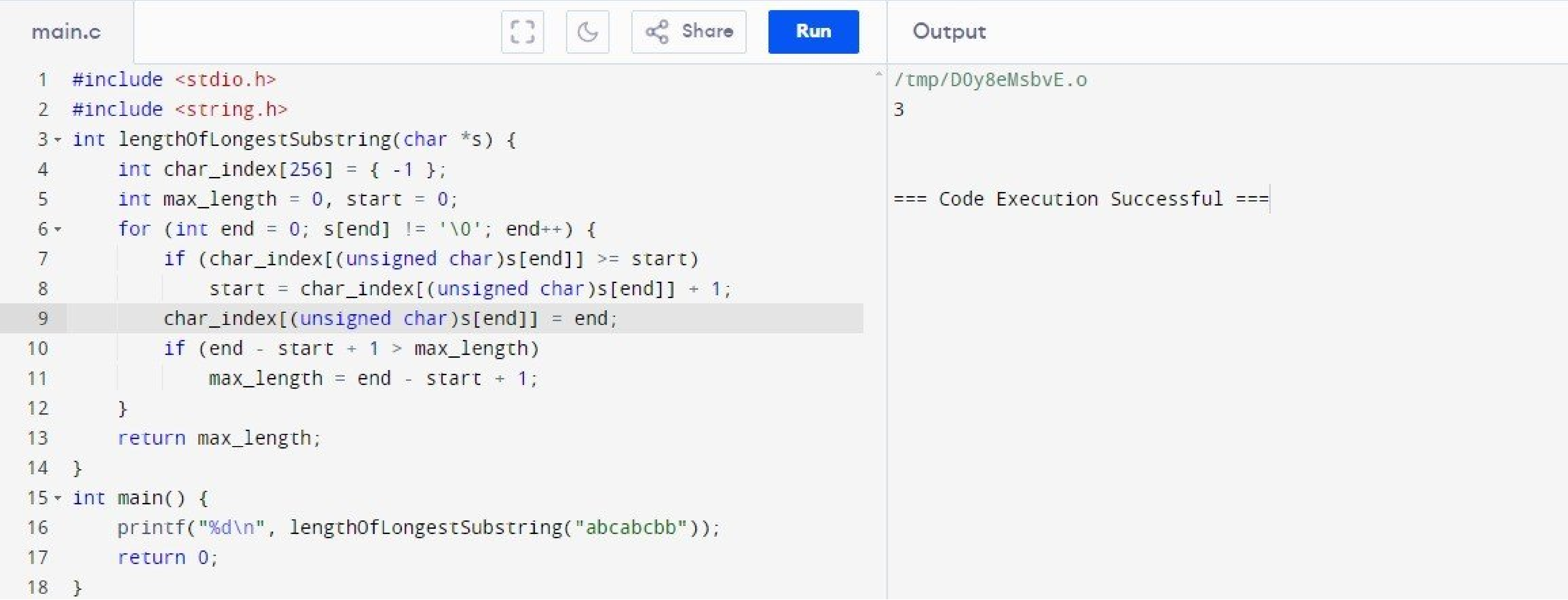


```c
#include <stdio.h>
#include <stdlib.h>

// Definition of the linked list node
typedef struct ListNode {
    int val;
    struct ListNode *next;
} ListNode;

// Function to create a new node
ListNode* createNode(int value) {
    ListNode* newNode = (ListNode*)malloc(sizeof(ListNode));
    newNode->val = value;
    newNode->next = NULL;
    return newNode;
}

// Function to add two numbers represented by linked lists
ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
    ListNode *dummyHead = createNode(0);
    ListNode *current = dummyHead;
    int carry = 0;

    while (l1 || l2 || carry) {
        int sum = carry;
        if (l1) {
            sum += l1->val;
            l1 = l1->next;
```

```
/tmp/BVj3HzH7KT.o
7 0 8


=== Code Execution Successful ===
```

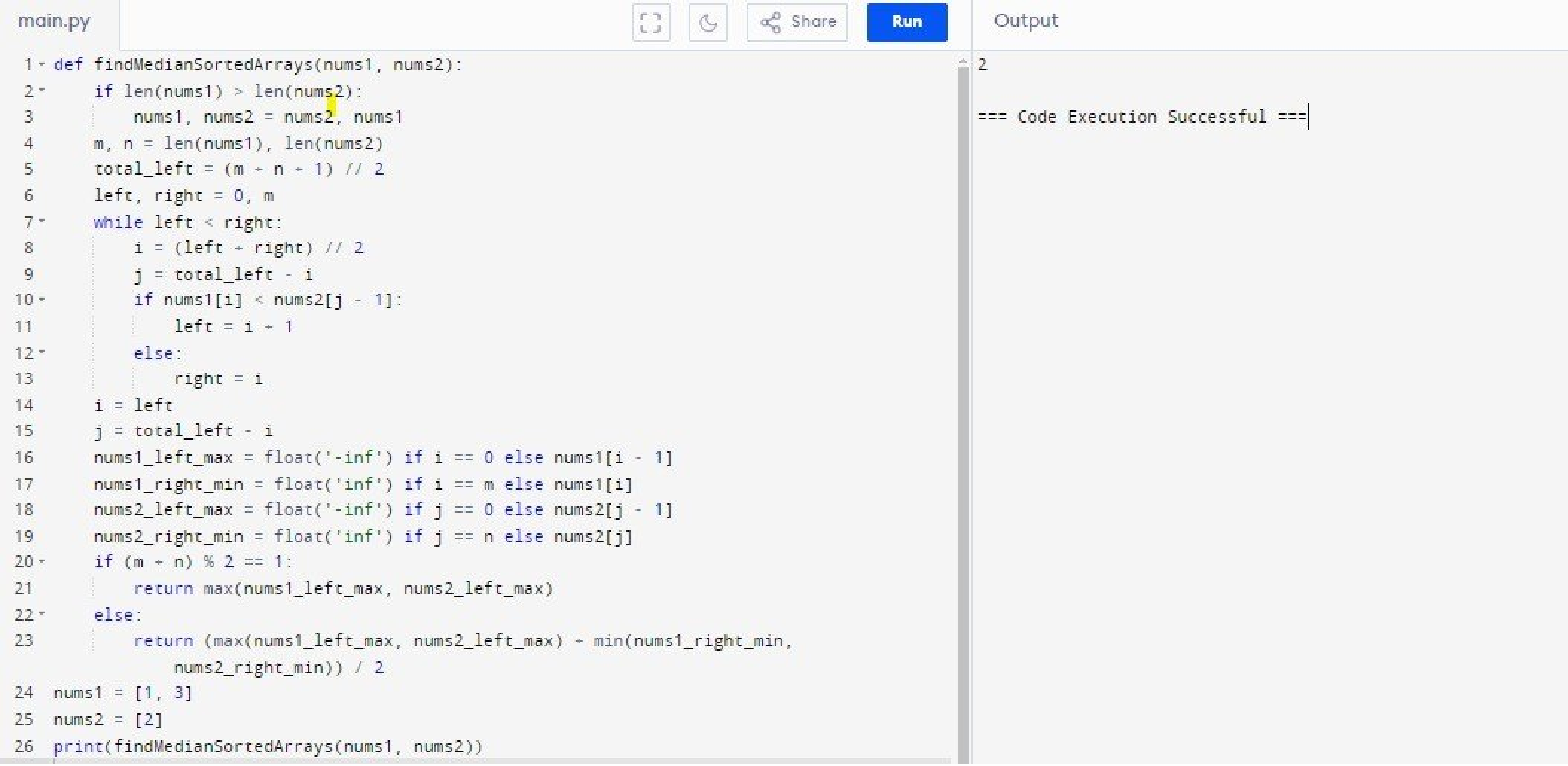


```
        if (l2) {
            sum += l2->val;
            l2 = l2->next;
        }
        carry = sum / 10;
        current->next = createNode(sum % 10);
        current = current->next;
    }

    return dummyHead->next;
}
void printList(ListNode* head) {
    while (head) {
        printf("%d ", head->val);
        head = head->next;
    }
    printf("\n");
}
int main() {
    ListNode* l1 = createNode(2);
    l1->next = createNode(4);
    l1->next->next = createNode(3);
    ListNode* l2 = createNode(5);
    l2->next = createNode(6);
    l2->next->next = createNode(4);
    ListNode* result = addTwoNumbers(l1, l2);
    printList(result);
    return 0;
```



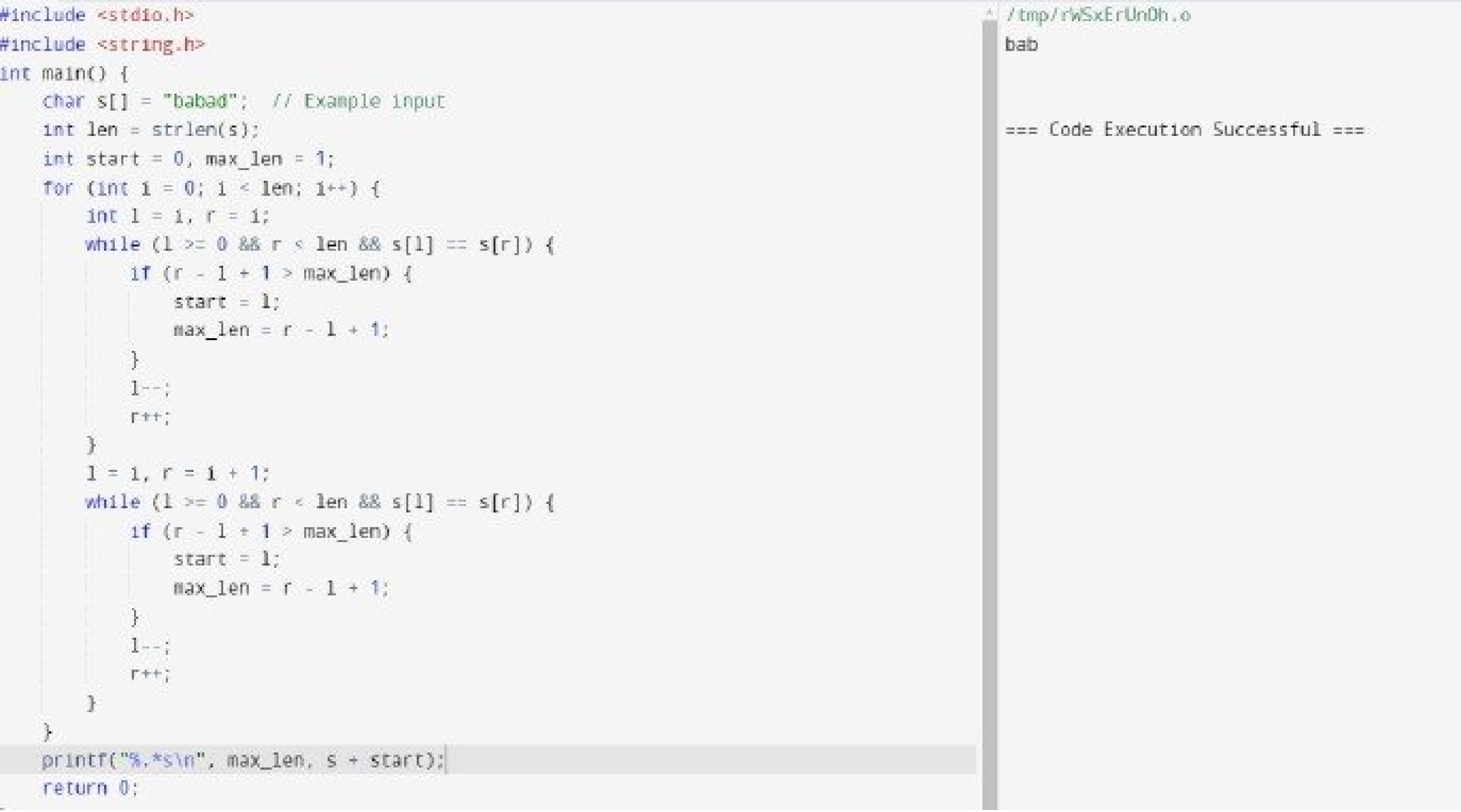
```c
#include <stdio.h>
#include <string.h>
int lengthOfLongestSubstring(char *s) {
    int char_index[256] = { -1 };
    int max_length = 0, start = 0;
    for (int end = 0; s[end] != '\0'; end++) {
        if (char_index[(unsigned char)s[end]] >= start)
            start = char_index[(unsigned char)s[end]] + 1;
        char_index[(unsigned char)s[end]] = end;
        if (end - start + 1 > max_length)
            max_length = end - start + 1;
    }
    return max_length;
}
int main() {
    printf("%d\n", lengthOfLongestSubstring("abcabcbb"));
    return 0;
}
```

```
/tmp/DOy8eMsbvE.o
3


=== Code Execution Successful ===
```



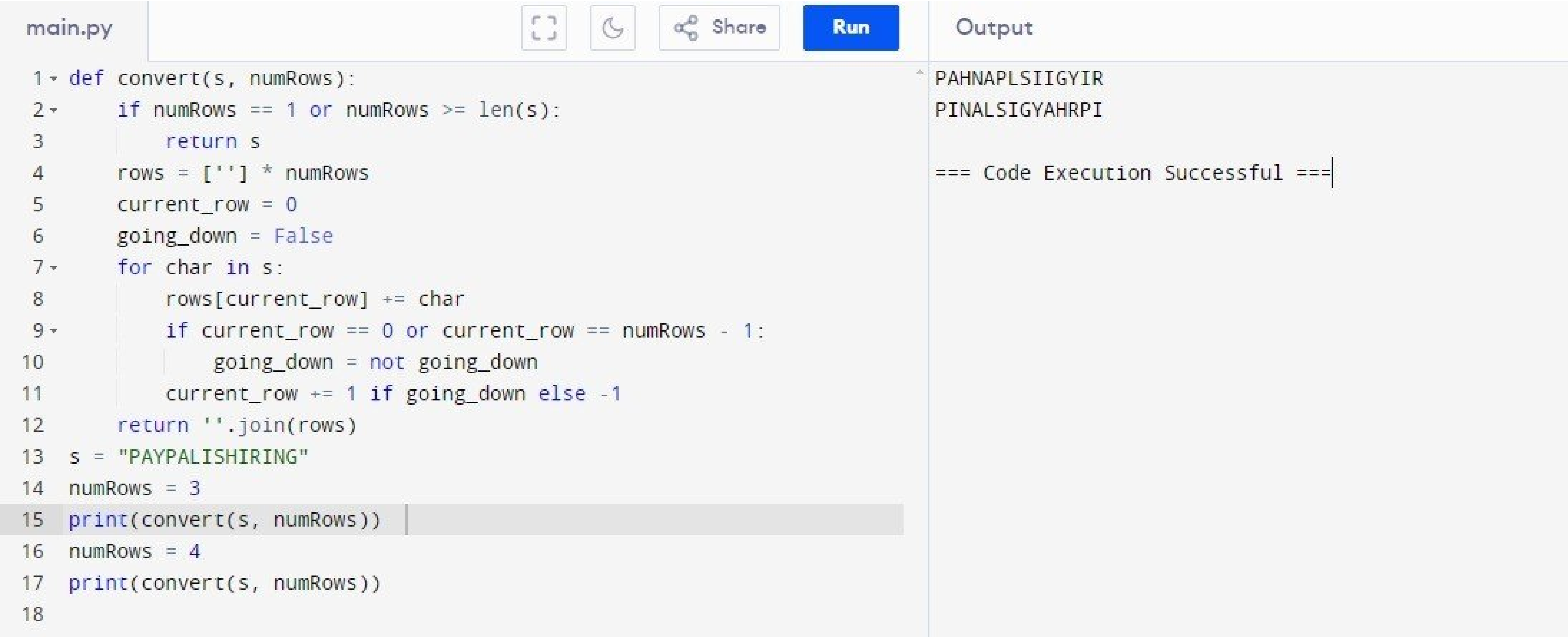
```python
def findMedianSortedArrays(nums1, nums2):
    if len(nums1) > len(nums2):
        nums1, nums2 = nums2, nums1
    m, n = len(nums1), len(nums2)
    total_left = (m + n + 1) // 2
    left, right = 0, m
    while left < right:
        i = (left + right) // 2
        j = total_left - i
        if nums1[i] < nums2[j - 1]:
            left = i + 1
        else:
            right = i
    i = left
    j = total_left - i
    nums1_left_max = float('-inf') if i == 0 else nums1[i - 1]
    nums1_right_min = float('inf') if i == m else nums1[i]
    nums2_left_max = float('-inf') if j == 0 else nums2[j - 1]
    nums2_right_min = float('inf') if j == n else nums2[j]
    if (m + n) % 2 == 1:
        return max(nums1_left_max, nums2_left_max)
    else:
        return (max(nums1_left_max, nums2_left_max) + min(nums1_right_min, nums2_right_min)) / 2
nums1 = [1, 3]
nums2 = [2]
print(findMedianSortedArrays(nums1, nums2))
```

```
2

=== Code Execution Successful ===
```



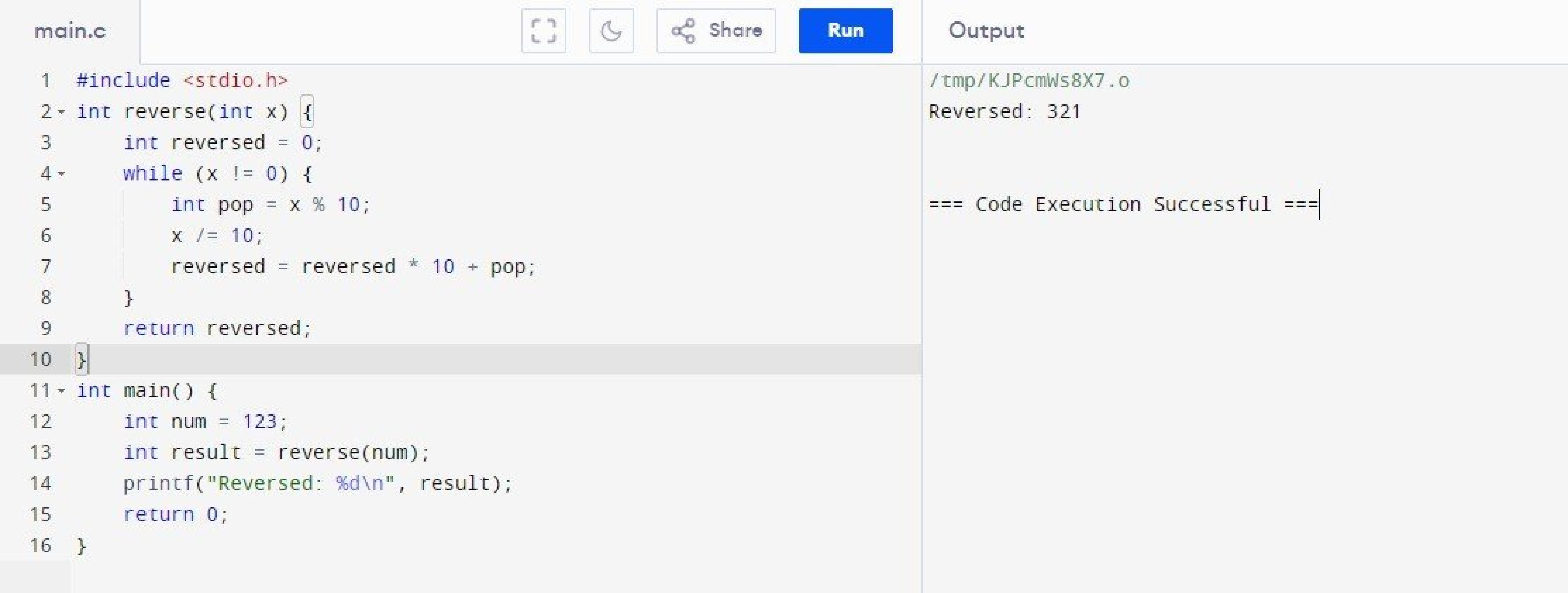
```c
#include <stdio.h>
#include <string.h>
int main() {
    char s[] = "babad";  // Example input
    int len = strlen(s);
    int start = 0, max_len = 1;
    for (int i = 0; i < len; i++) {
        int l = i, r = i;
        while (l >= 0 && r < len && s[l] == s[r]) {
            if (r - l + 1 > max_len) {
                start = l;
                max_len = r - l + 1;
            }
            l--;
            r++;
        }
        l = i, r = i + 1;
        while (l >= 0 && r < len && s[l] == s[r]) {
            if (r - l + 1 > max_len) {
                start = l;
                max_len = r - l + 1;
            }
            l--;
            r++;
        }
    }
    printf("%.*s\n", max_len, s + start);
    return 0;
```

```
/tmp/rWSxErUnOh.o
bab


=== Code Execution Successful ===
```



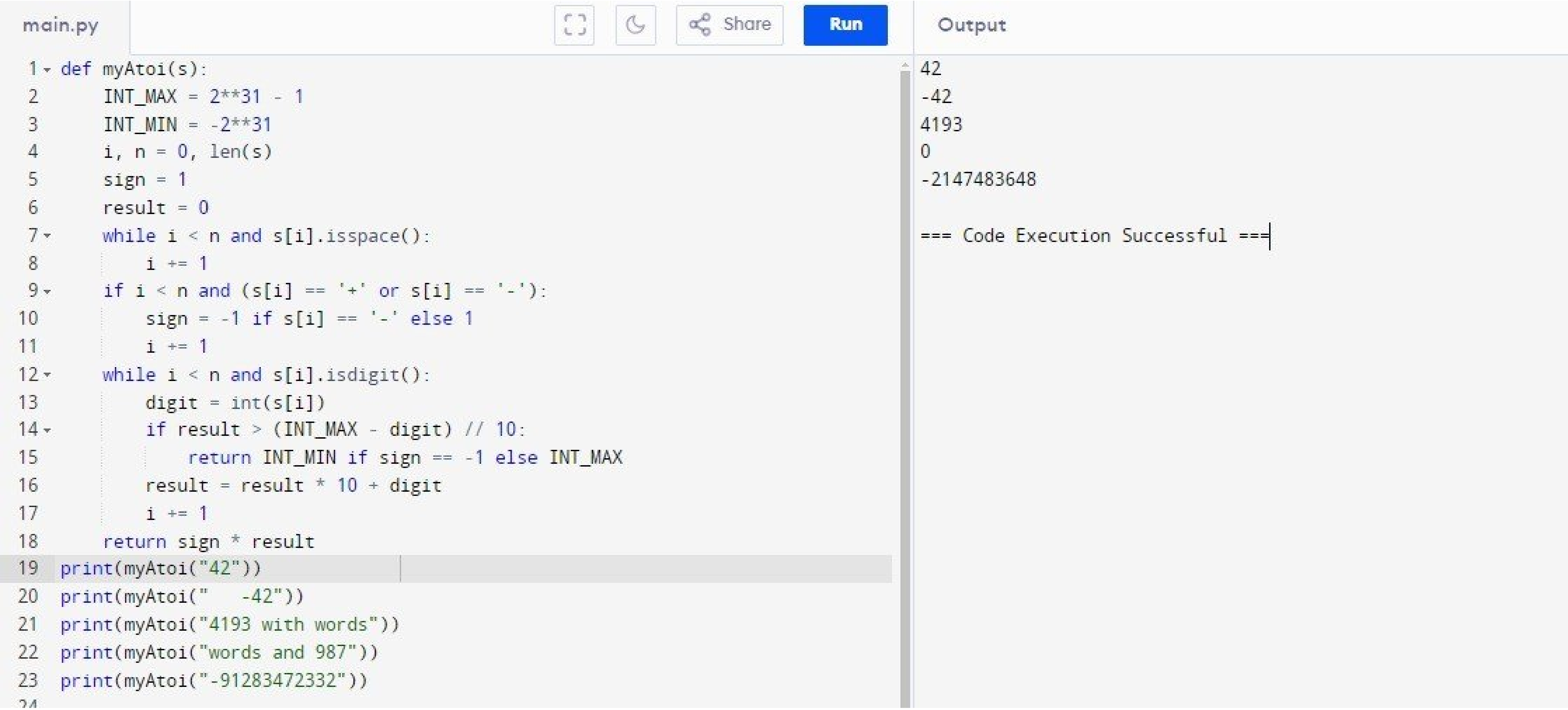
```python
def convert(s, numRows):
    if numRows == 1 or numRows >= len(s):
        return s
    rows = [''] * numRows
    current_row = 0
    going_down = False
    for char in s:
        rows[current_row] += char
        if current_row == 0 or current_row == numRows - 1:
            going_down = not going_down
        current_row += 1 if going_down else -1
    return ''.join(rows)
s = "PAYPALISHIRING"
numRows = 3
print(convert(s, numRows))
numRows = 4
print(convert(s, numRows))
```

```
PAHNAPLSIIGYIR
PINALSIGYAHRPI

=== Code Execution Successful ===
```



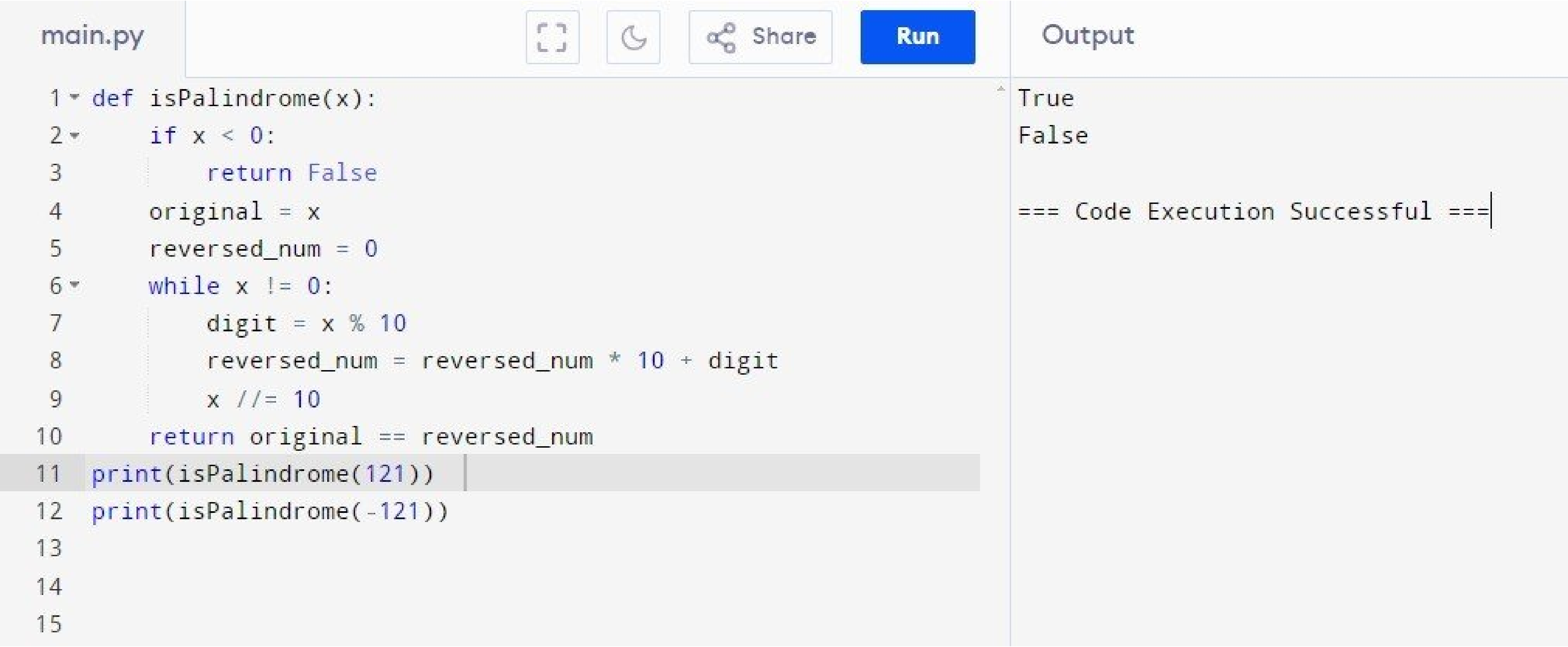
```c
#include <stdio.h>
int reverse(int x) {
    int reversed = 0;
    while (x != 0) {
        int pop = x % 10;
        x /= 10;
        reversed = reversed * 10 + pop;
    }
    return reversed;
}
int main() {
    int num = 123;
    int result = reverse(num);
    printf("Reversed: %d\n", result);
    return 0;
}
```

```
/tmp/KJPcmWs8X7.o
Reversed: 321


=== Code Execution Successful ===
```



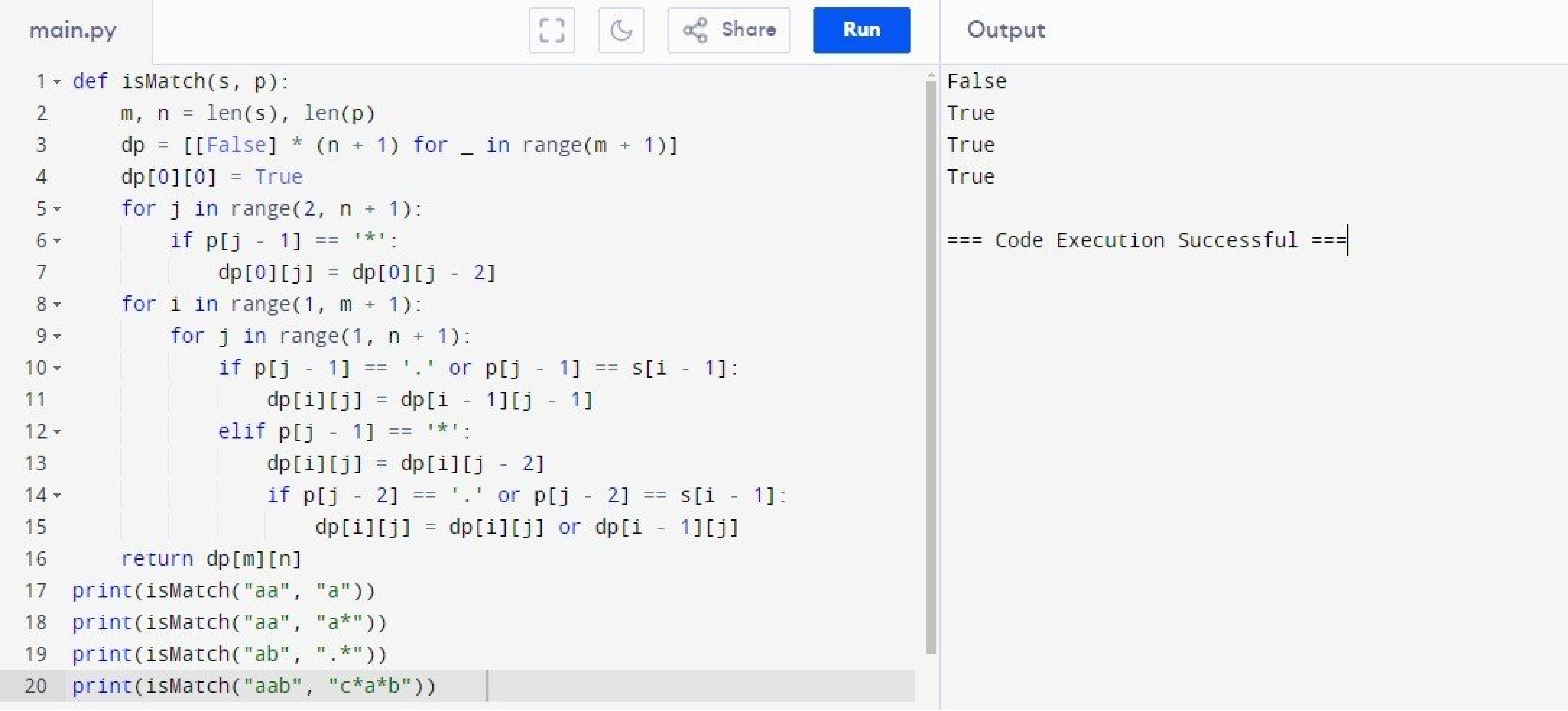
```python
def myAtoi(s):
    INT_MAX = 2**31 - 1
    INT_MIN = -2**31
    i, n = 0, len(s)
    sign = 1
    result = 0
    while i < n and s[i].isspace():
        i += 1
    if i < n and (s[i] == '+' or s[i] == '-'):
        sign = -1 if s[i] == '-' else 1
        i += 1
    while i < n and s[i].isdigit():
        digit = int(s[i])
        if result > (INT_MAX - digit) // 10:
            return INT_MIN if sign == -1 else INT_MAX
        result = result * 10 + digit
        i += 1
    return sign * result
print(myAtoi("42"))
print(myAtoi("   -42"))
print(myAtoi("4193 with words"))
print(myAtoi("words and 987"))
print(myAtoi("-91283472332"))
```

```
42
-42
4193
0
-2147483648

=== Code Execution Successful ===
```



```python
def isPalindrome(x):
    if x < 0:
        return False
    original = x
    reversed_num = 0
    while x != 0:
        digit = x % 10
        reversed_num = reversed_num * 10 + digit
        x //= 10
    return original == reversed_num
print(isPalindrome(121))
print(isPalindrome(-121))
```

```
True
False

=== Code Execution Successful ===
```



```python
def isMatch(s, p):
    m, n = len(s), len(p)
    dp = [[False] * (n + 1) for _ in range(m + 1)]
    dp[0][0] = True
    for j in range(2, n + 1):
        if p[j - 1] == '*':
            dp[0][j] = dp[0][j - 2]
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if p[j - 1] == '.' or p[j - 1] == s[i - 1]:
                dp[i][j] = dp[i - 1][j - 1]
            elif p[j - 1] == '*':
                dp[i][j] = dp[i][j - 2]
                if p[j - 2] == '.' or p[j - 2] == s[i - 1]:
                    dp[i][j] = dp[i][j] or dp[i - 1][j]
    return dp[m][n]
print(isMatch("aa", "a"))
print(isMatch("aa", "a*"))
print(isMatch("ab", ".*"))
print(isMatch("aab", "c*a*b"))
```

```
False
True
True
True

=== Code Execution Successful ===
```