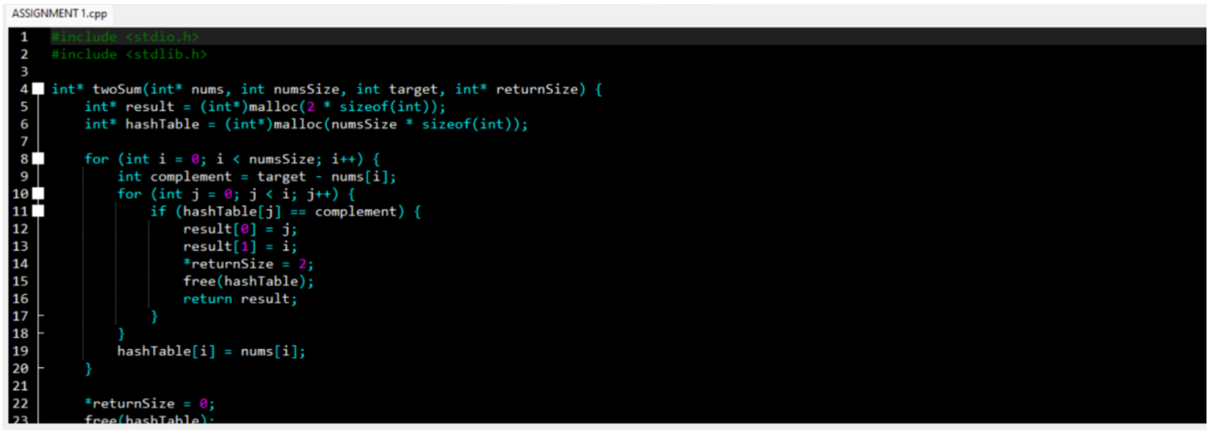
ASSIGNMENT 1

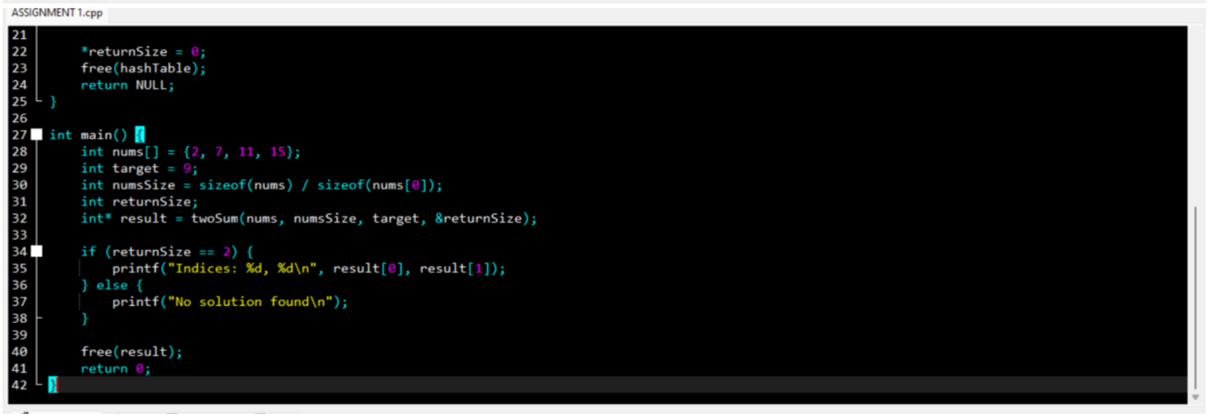
1)

CODE:



ASSIGNMENT 1.cpp

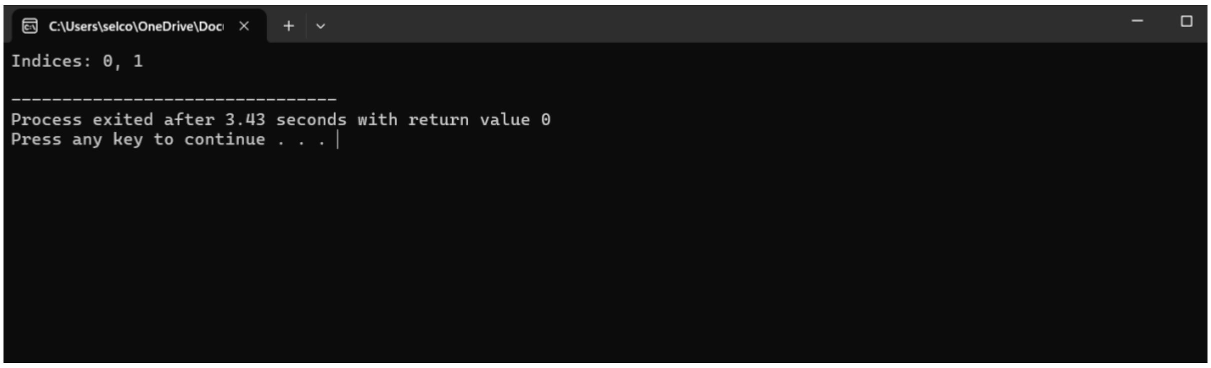
```c
#include <stdio.h>
#include <stdlib.h>

int* twoSum(int* nums, int numsSize, int target, int* returnSize) {
    int* result = (int*)malloc(2 * sizeof(int));
    int* hashTable = (int*)malloc(numsSize * sizeof(int));

    for (int i = 0; i < numsSize; i++) {
        int complement = target - nums[i];
        for (int j = 0; j < i; j++) {
            if (hashTable[j] == complement) {
                result[0] = j;
                result[1] = i;
                *returnSize = 2;
                free(hashTable);
                return result;
            }
        }
        hashTable[i] = nums[i];
    }

    *returnSize = 0;
    free(hashTable);
```



ASSIGNMENT 1.cpp

```c

    *returnSize = 0;
    free(hashTable);
    return NULL;
}

int main() {
    int nums[] = {2, 7, 11, 15};
    int target = 9;
    int numsSize = sizeof(nums) / sizeof(nums[0]);
    int returnSize;
    int* result = twoSum(nums, numsSize, target, &returnSize);

    if (returnSize == 2) {
        printf("Indices: %d, %d\n", result[0], result[1]);
    } else {
        printf("No solution found\n");
    }

    free(result);
    return 0;
}
```

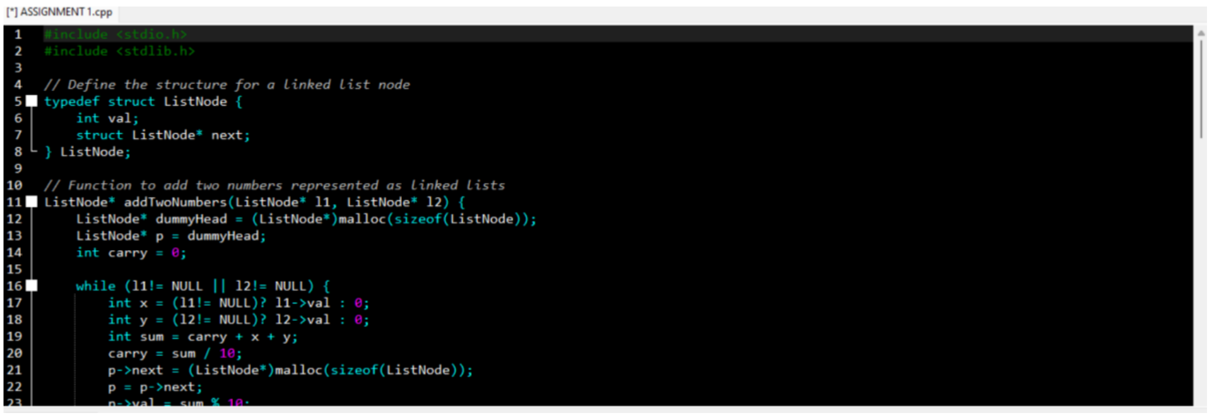
OUTPUT:



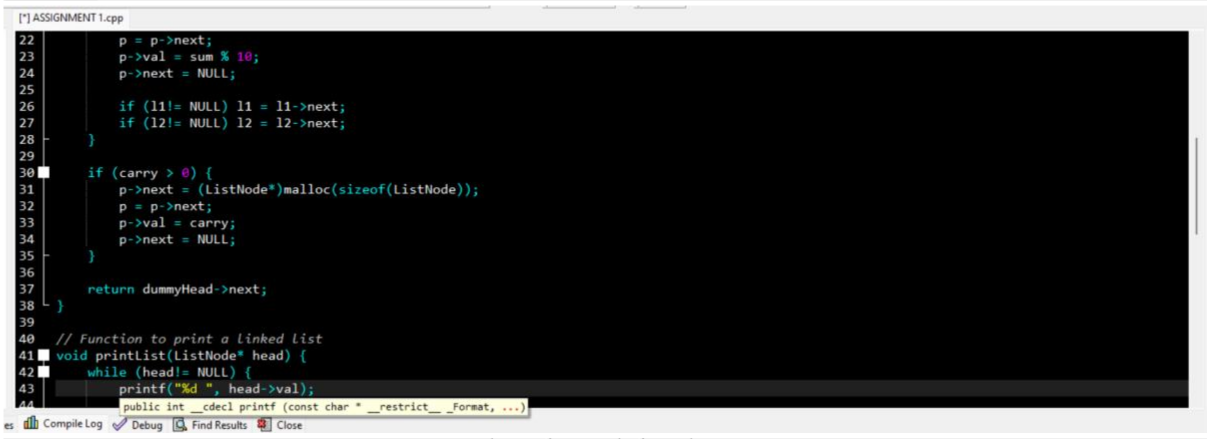
C:\Users\selco\OneDrive\Doc

```
Indices: 0, 1

--------------------------------
Process exited after 3.43 seconds with return value 0
Press any key to continue . . .
```

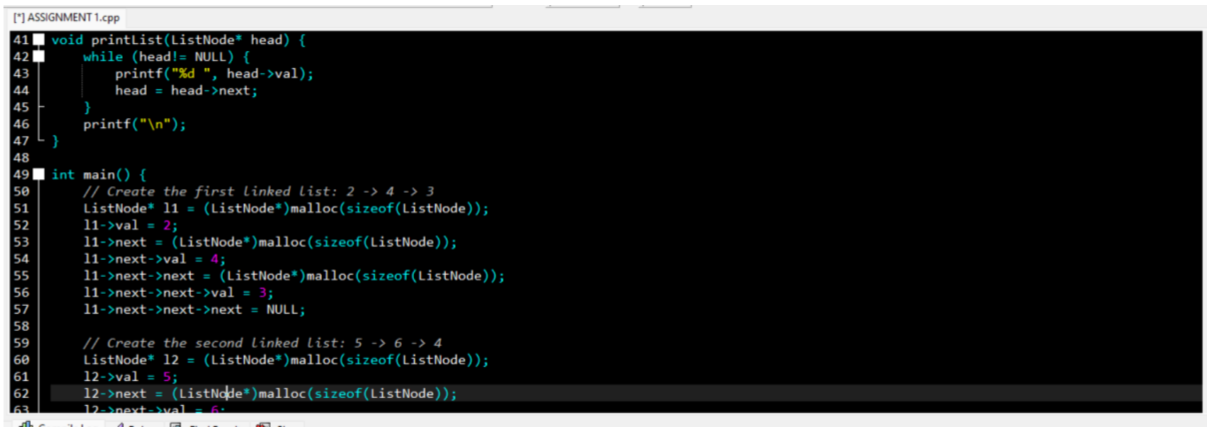
2)
CODE:



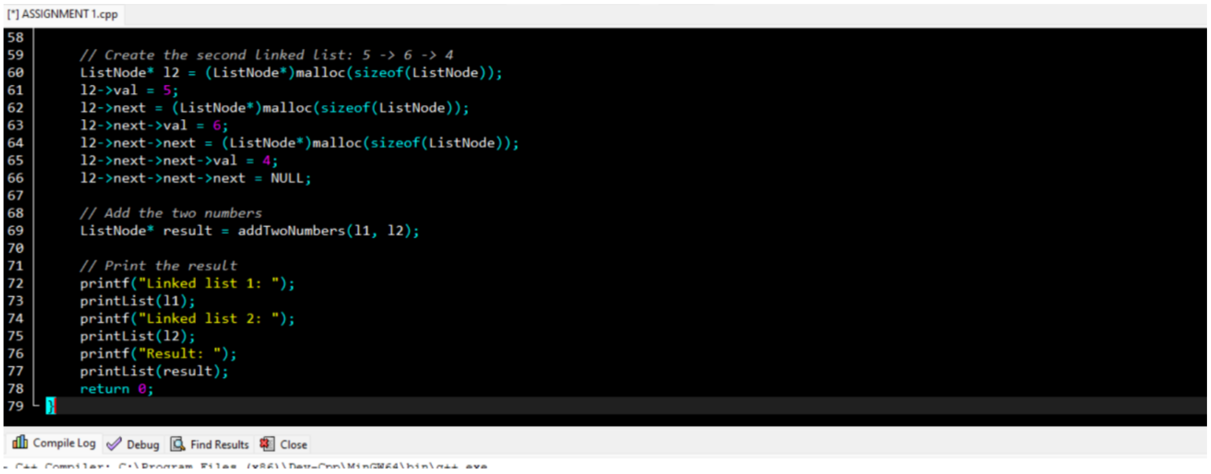
```c
#include <stdio.h>
#include <stdlib.h>

// Define the structure for a linked list node
typedef struct ListNode {
    int val;
    struct ListNode* next;
} ListNode;

// Function to add two numbers represented as linked lists
ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
    ListNode* dummyHead = (ListNode*)malloc(sizeof(ListNode));
    ListNode* p = dummyHead;
    int carry = 0;

    while (l1!= NULL || l2!= NULL) {
        int x = (l1!= NULL)? l1->val : 0;
        int y = (l2!= NULL)? l2->val : 0;
        int sum = carry + x + y;
        carry = sum / 10;
        p->next = (ListNode*)malloc(sizeof(ListNode));
        p = p->next;
        p->val = sum % 10;
```


[*] ASSIGNMENT 1.cpp
public int __cdecl printf (const char * __restrict__ _Format, ...)
Compile Log
Debug
Find Results
Close

```c
        p = p->next;
        p->val = sum % 10;
        p->next = NULL;

        if (l1!= NULL) l1 = l1->next;
        if (l2!= NULL) l2 = l2->next;
    }

    if (carry > 0) {
        p->next = (ListNode*)malloc(sizeof(ListNode));
        p = p->next;
        p->val = carry;
        p->next = NULL;
    }

    return dummyHead->next;
}

// Function to print a linked list
void printList(ListNode* head) {
    while (head!= NULL) {
        printf("%d ", head->val);
```



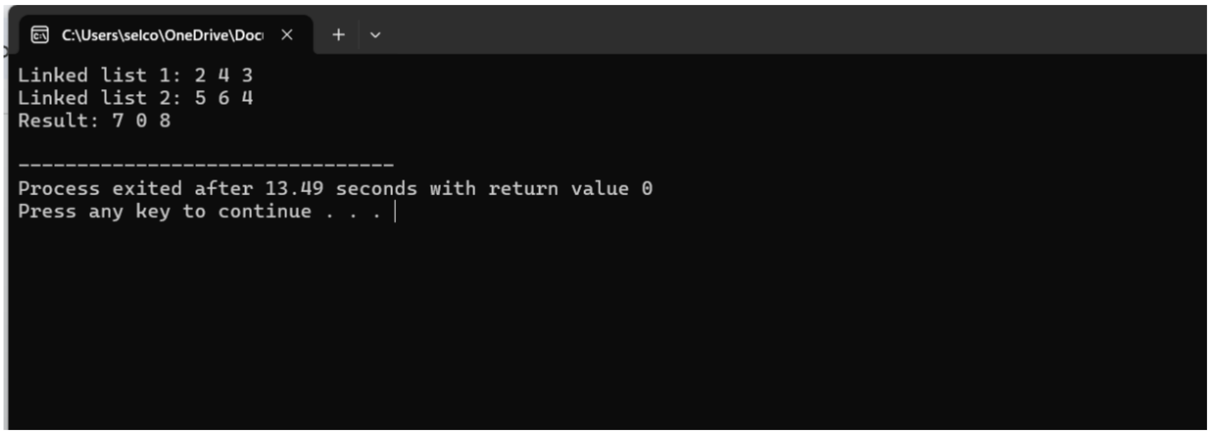
```c
void printList(ListNode* head) {
    while (head!= NULL) {
        printf("%d ", head->val);
        head = head->next;
    }
    printf("\n");
}

int main() {
    // Create the first linked list: 2 -> 4 -> 3
    ListNode* l1 = (ListNode*)malloc(sizeof(ListNode));
    l1->val = 2;
    l1->next = (ListNode*)malloc(sizeof(ListNode));
    l1->next->val = 4;
    l1->next->next = (ListNode*)malloc(sizeof(ListNode));
    l1->next->next->val = 3;
    l1->next->next->next = NULL;

    // Create the second linked list: 5 -> 6 -> 4
    ListNode* l2 = (ListNode*)malloc(sizeof(ListNode));
    l2->val = 5;
    l2->next = (ListNode*)malloc(sizeof(ListNode));
    l2->next->val = 6;
```



```c

    // Create the second linked list: 5 -> 6 -> 4
    ListNode* l2 = (ListNode*)malloc(sizeof(ListNode));
    l2->val = 5;
    l2->next = (ListNode*)malloc(sizeof(ListNode));
    l2->next->val = 6;
    l2->next->next = (ListNode*)malloc(sizeof(ListNode));
    l2->next->next->val = 4;
    l2->next->next->next = NULL;

    // Add the two numbers
    ListNode* result = addTwoNumbers(l1, l2);

    // Print the result
    printf("Linked list 1: ");
    printList(l1);
    printf("Linked list 2: ");
    printList(l2);
    printf("Result: ");
    printList(result);
    return 0;
}
```

```
- C++ Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\g++.exe
```

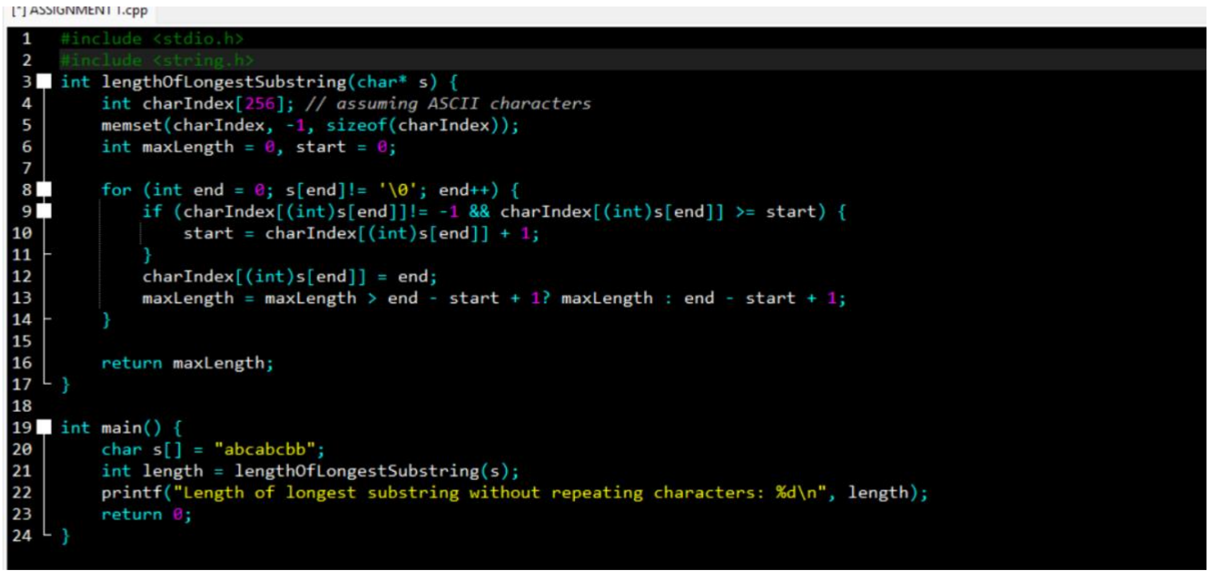
OUTPUT:



```
Linked list 1: 2 4 3
Linked list 2: 5 6 4
Result: 7 0 8

--------------------------------
Process exited after 13.49 seconds with return value 0
Press any key to continue . . .
```

3)

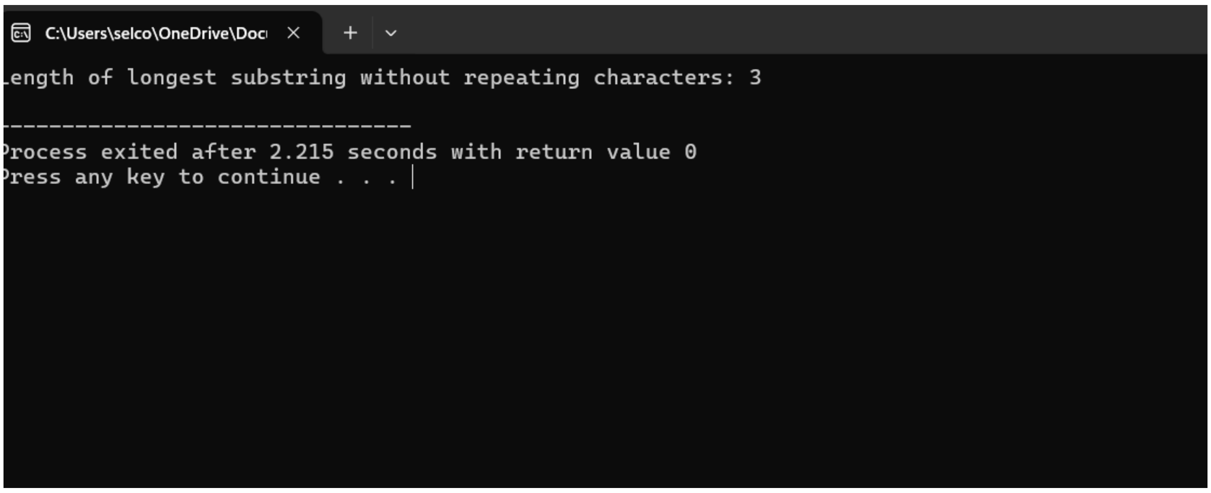
CODE:



[*] ASSIGNMENT 1.cpp

```c
#include <stdio.h>
#include <string.h>
int lengthOfLongestSubstring(char* s) {
    int charIndex[256]; // assuming ASCII characters
    memset(charIndex, -1, sizeof(charIndex));
    int maxLength = 0, start = 0;

    for (int end = 0; s[end]!= '\0'; end++) {
        if (charIndex[(int)s[end]]!= -1 && charIndex[(int)s[end]] >= start) {
            start = charIndex[(int)s[end]] + 1;
        }
        charIndex[(int)s[end]] = end;
        maxLength = maxLength > end - start + 1? maxLength : end - start + 1;
    }

    return maxLength;
}

int main() {
    char s[] = "abcabcbb";
    int length = lengthOfLongestSubstring(s);
    printf("Length of longest substring without repeating characters: %d\n", length);
    return 0;
}
```

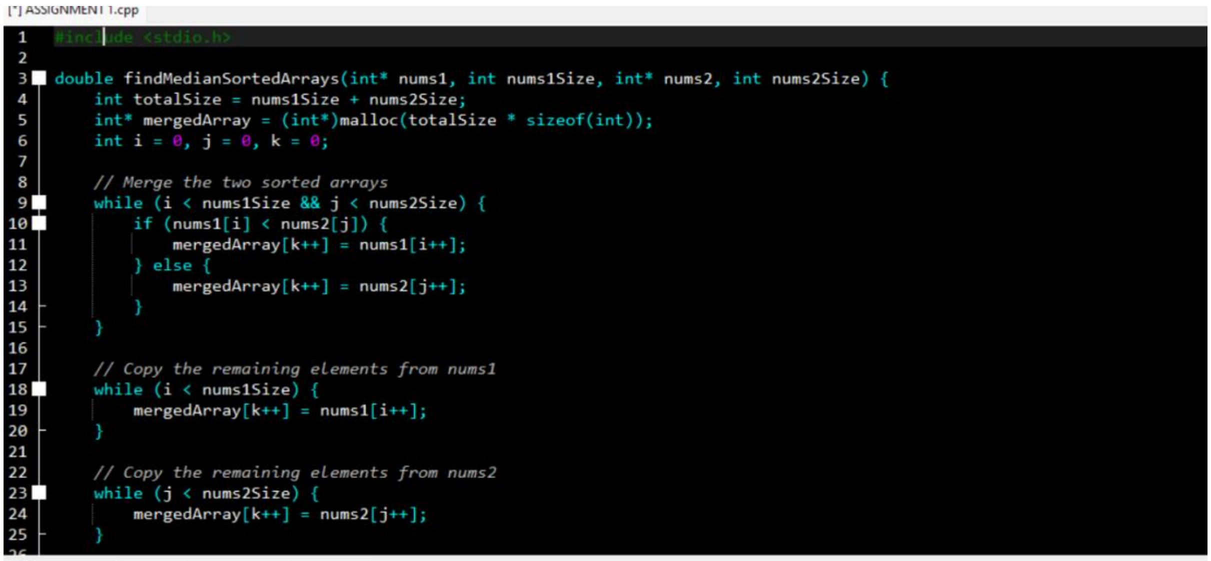
OUTPUT:



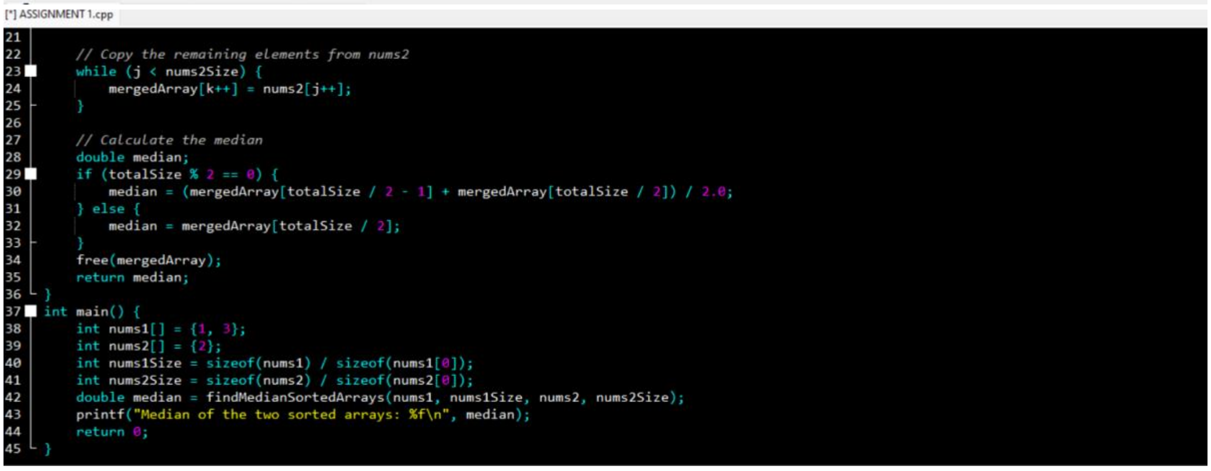
C:\Users\selco\OneDrive\Doc

```
Length of longest substring without repeating characters: 3

--------------------------------
Process exited after 2.215 seconds with return value 0
Press any key to continue . . .
```

4)
CODE:

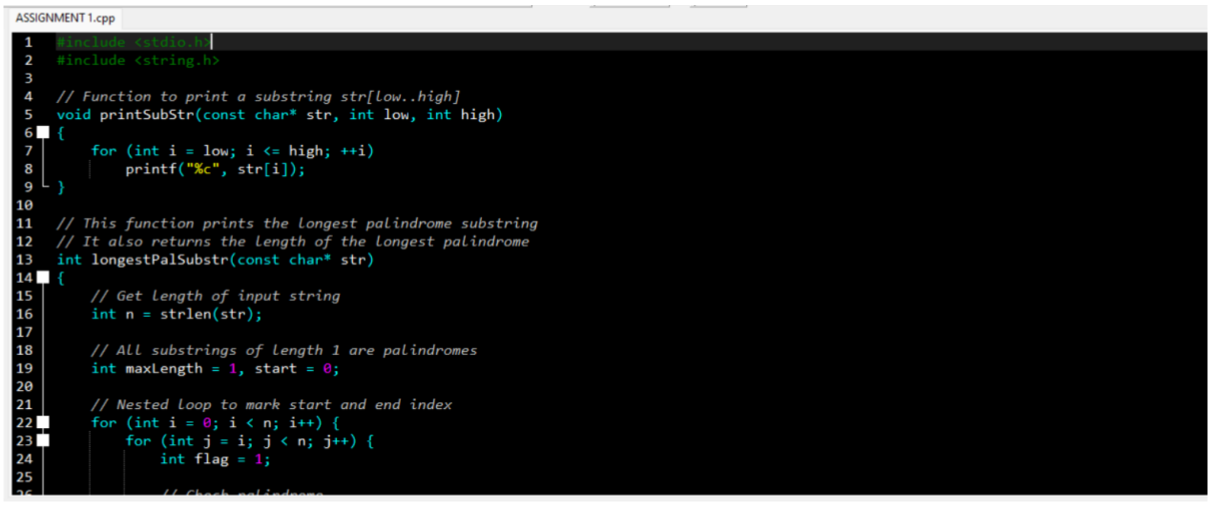

[*] ASSIGNMENT 1.cpp

```c
#include <stdio.h>

double findMedianSortedArrays(int* nums1, int nums1Size, int* nums2, int nums2Size) {
    int totalSize = nums1Size + nums2Size;
    int* mergedArray = (int*)malloc(totalSize * sizeof(int));
    int i = 0, j = 0, k = 0;

    // Merge the two sorted arrays
    while (i < nums1Size && j < nums2Size) {
        if (nums1[i] < nums2[j]) {
            mergedArray[k++] = nums1[i++];
        } else {
            mergedArray[k++] = nums2[j++];
        }
    }

    // Copy the remaining elements from nums1
    while (i < nums1Size) {
        mergedArray[k++] = nums1[i++];
    }

    // Copy the remaining elements from nums2
    while (j < nums2Size) {
        mergedArray[k++] = nums2[j++];
    }
```

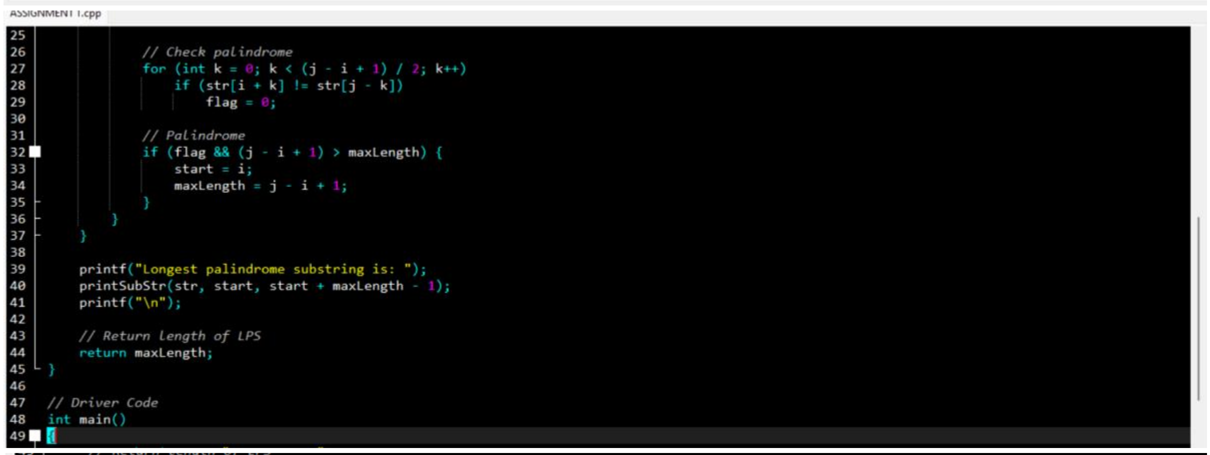

[*] ASSIGNMENT 1.cpp

```c

    // Copy the remaining elements from nums2
    while (j < nums2Size) {
        mergedArray[k++] = nums2[j++];
    }

    // Calculate the median
    double median;
    if (totalSize % 2 == 0) {
        median = (mergedArray[totalSize / 2 - 1] + mergedArray[totalSize / 2]) / 2.0;
    } else {
        median = mergedArray[totalSize / 2];
    }
    free(mergedArray);
    return median;
}
int main() {
    int nums1[] = {1, 3};
    int nums2[] = {2};
    int nums1Size = sizeof(nums1) / sizeof(nums1[0]);
    int nums2Size = sizeof(nums2) / sizeof(nums2[0]);
    double median = findMedianSortedArrays(nums1, nums1Size, nums2, nums2Size);
    printf("Median of the two sorted arrays: %f\n", median);
    return 0;
}
```

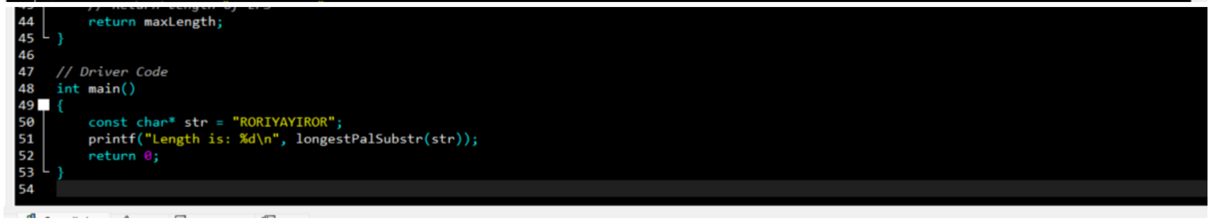
5)
CODE:


ASSIGNMENT 1.cpp

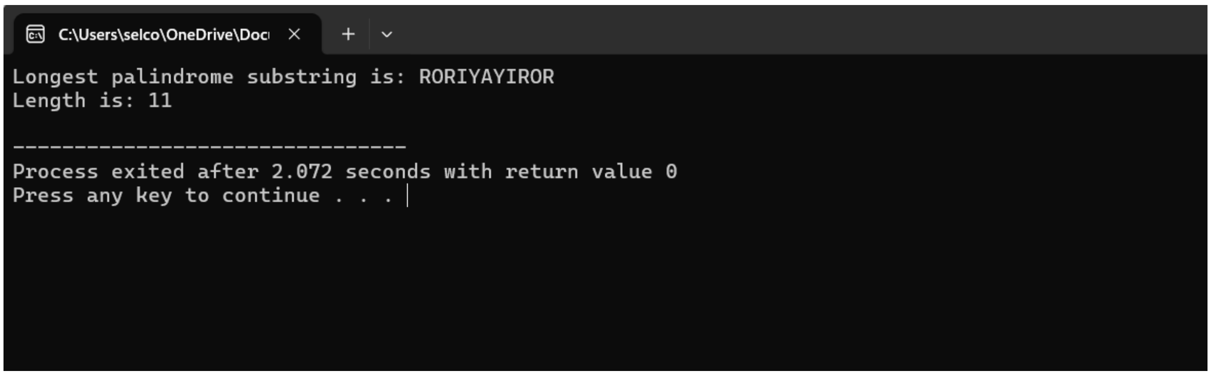
```
#include <stdio.h>
#include <string.h>

// Function to print a substring str[low..high]
void printSubStr(const char* str, int low, int high)
{
    for (int i = low; i <= high; ++i)
        printf("%c", str[i]);
}

// This function prints the longest palindrome substring
// It also returns the length of the longest palindrome
int longestPalSubstr(const char* str)
{
    // Get length of input string
    int n = strlen(str);

    // All substrings of length 1 are palindromes
    int maxLength = 1, start = 0;

    // Nested loop to mark start and end index
    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            int flag = 1;

```


ASSIGNMENT 1.cpp

```

            // Check palindrome
            for (int k = 0; k < (j - i + 1) / 2; k++)
                if (str[i + k] != str[j - k])
                    flag = 0;

            // Palindrome
            if (flag && (j - i + 1) > maxLength) {
                start = i;
                maxLength = j - i + 1;
            }
        }
    }

    printf("Longest palindrome substring is: ");
    printSubStr(str, start, start + maxLength - 1);
    printf("\n");

    // Return length of LPS
    return maxLength;
}

// Driver Code
int main()
{
```

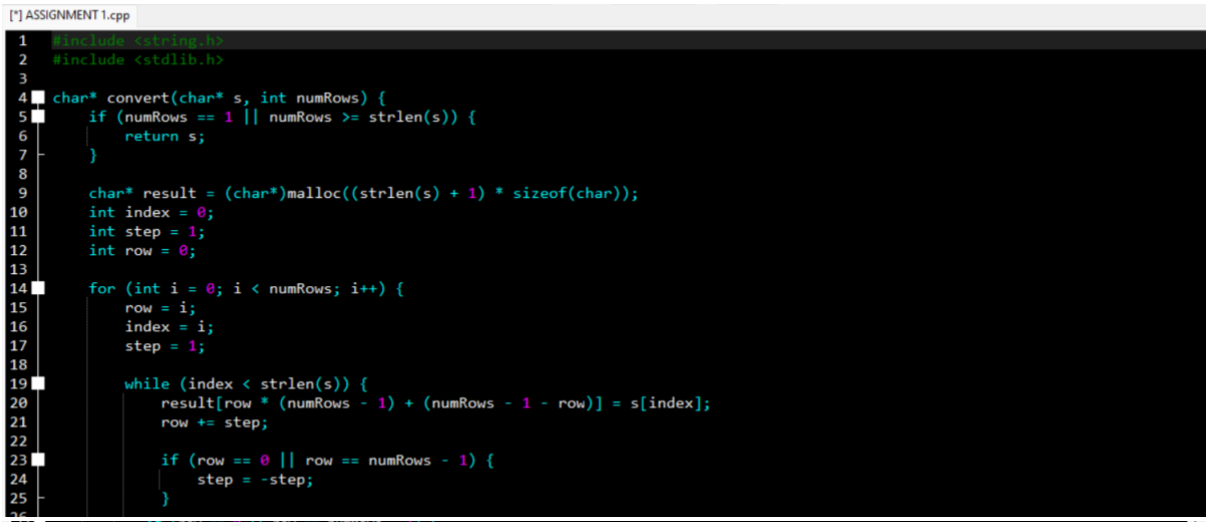

```
    return maxLength;
}

// Driver Code
int main()
{
    const char* str = "RORIYAYIROR";
    printf("Length is: %d\n", longestPalSubstr(str));
    return 0;
}

```

OUTPUT:

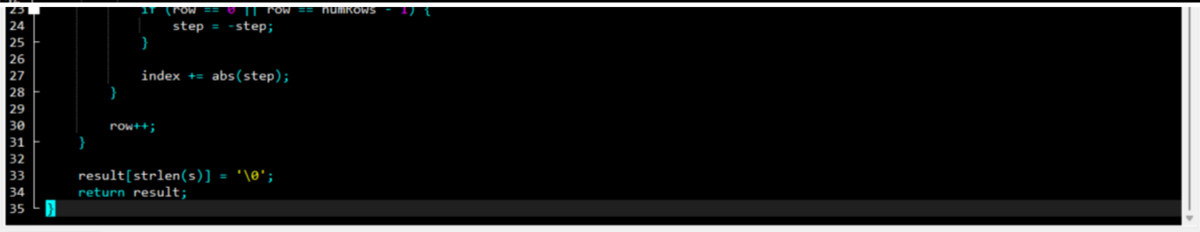

C:\Users\selco\OneDrive\Doc

```
Longest palindrome substring is: RORIYAYIROR
Length is: 11

--------------------------------
Process exited after 2.072 seconds with return value 0
Press any key to continue . . .
```

6)
CODE:


[*] ASSIGNMENT 1.cpp

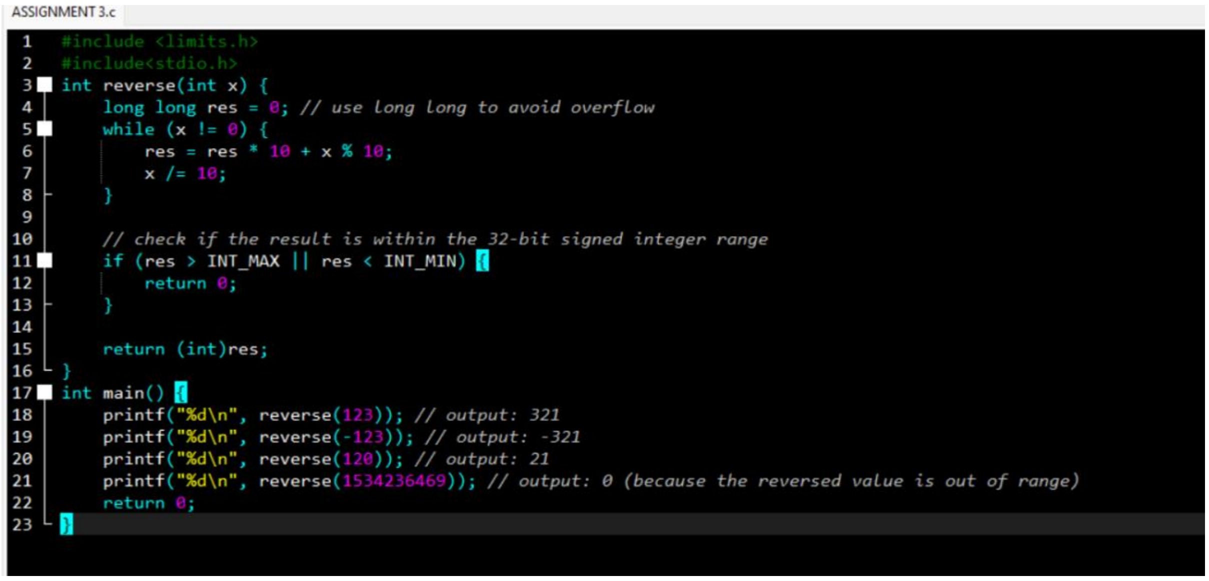
```c
#include <string.h>
#include <stdlib.h>

char* convert(char* s, int numRows) {
    if (numRows == 1 || numRows >= strlen(s)) {
        return s;
    }

    char* result = (char*)malloc((strlen(s) + 1) * sizeof(char));
    int index = 0;
    int step = 1;
    int row = 0;

    for (int i = 0; i < numRows; i++) {
        row = i;
        index = i;
        step = 1;

        while (index < strlen(s)) {
            result[row * (numRows - 1) + (numRows - 1 - row)] = s[index];
            row += step;

            if (row == 0 || row == numRows - 1) {
                step = -step;
            }
```



```c
            if (row == 0 || row == numRows - 1) {
                step = -step;
            }
```

```c

            index += abs(step);
        }

        row++;
    }

    result[strlen(s)] = '\0';
    return result;
}
```

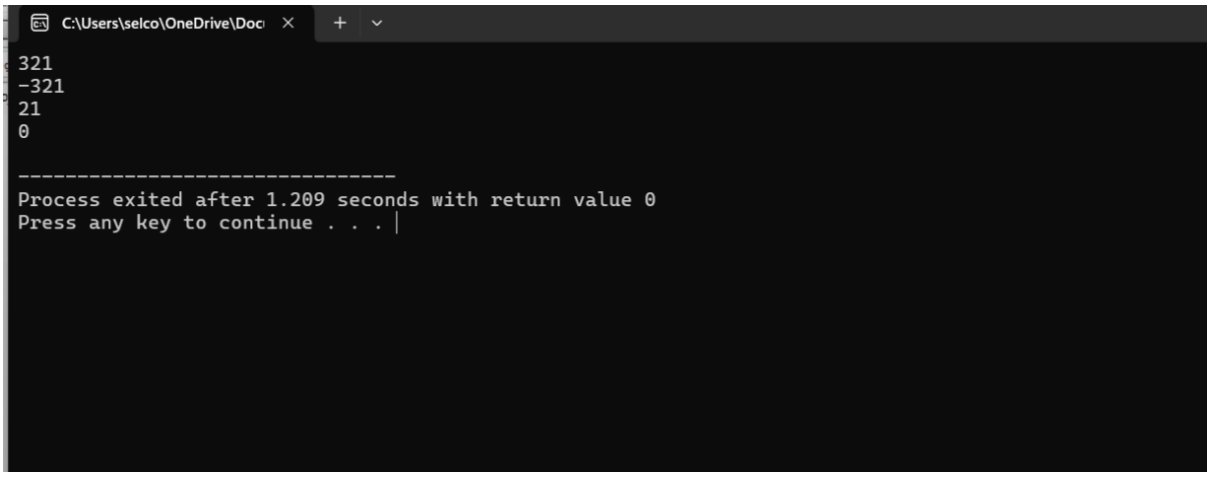
7)
CODE:



ASSIGNMENT 3.c

```c
#include <limits.h>
#include<stdio.h>
int reverse(int x) {
    long long res = 0; // use long long to avoid overflow
    while (x != 0) {
        res = res * 10 + x % 10;
        x /= 10;
    }

    // check if the result is within the 32-bit signed integer range
    if (res > INT_MAX || res < INT_MIN) {
        return 0;
    }

    return (int)res;
}
int main() {
    printf("%d\n", reverse(123)); // output: 321
    printf("%d\n", reverse(-123)); // output: -321
    printf("%d\n", reverse(120)); // output: 21
    printf("%d\n", reverse(1534236469)); // output: 0 (because the reversed value is out of range)
    return 0;
}
```

OUTPUT:

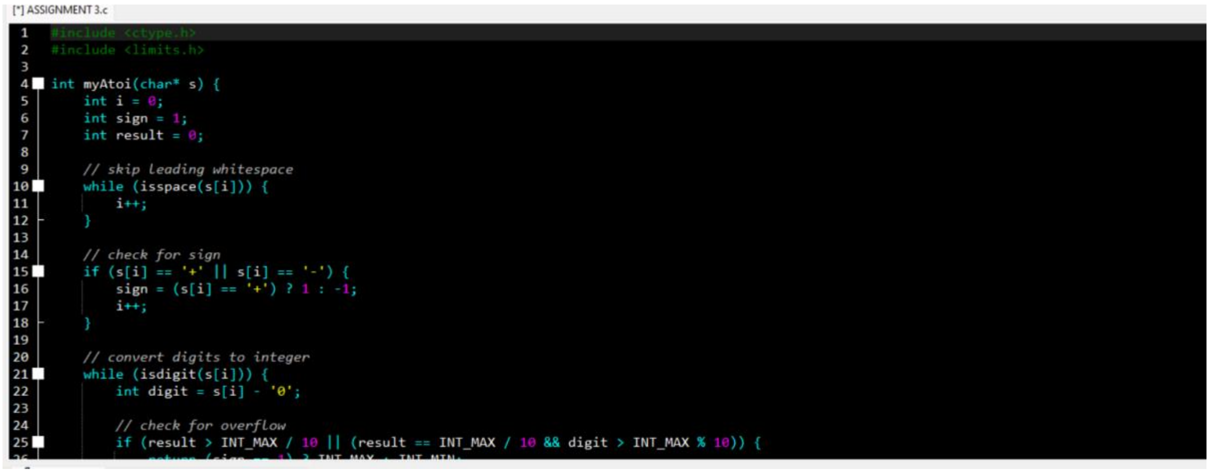


C:\Users\selco\OneDrive\Doc

```
321
-321
21
0

--------------------------------
Process exited after 1.209 seconds with return value 0
Press any key to continue . . .
```

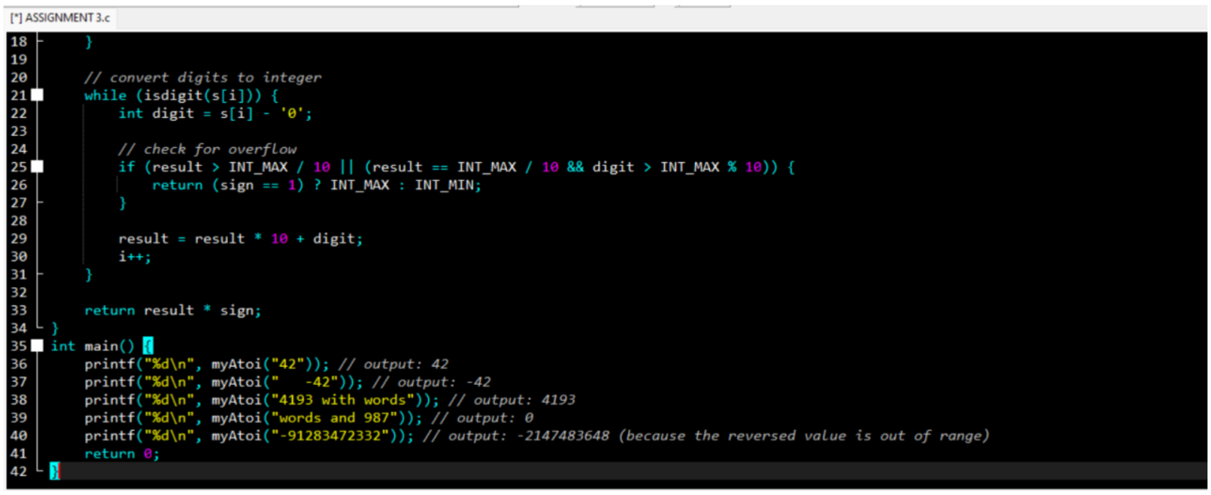
8)

CODE;



[*] ASSIGNMENT 3.c

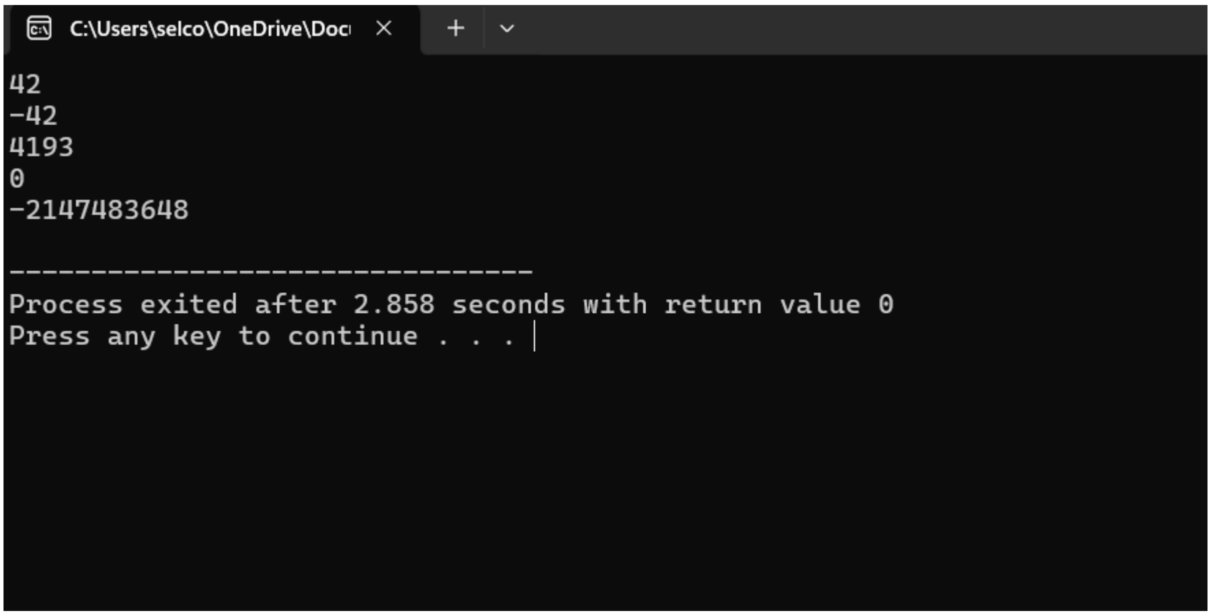
```c
#include <ctype.h>
#include <limits.h>

int myAtoi(char* s) {
    int i = 0;
    int sign = 1;
    int result = 0;

    // skip leading whitespace
    while (isspace(s[i])) {
        i++;
    }

    // check for sign
    if (s[i] == '+' || s[i] == '-') {
        sign = (s[i] == '+') ? 1 : -1;
        i++;
    }

    // convert digits to integer
    while (isdigit(s[i])) {
        int digit = s[i] - '0';

        // check for overflow
        if (result > INT_MAX / 10 || (result == INT_MAX / 10 && digit > INT_MAX % 10)) {
```



[*] ASSIGNMENT 3.c

```c
    }

    // convert digits to integer
    while (isdigit(s[i])) {
        int digit = s[i] - '0';

        // check for overflow
        if (result > INT_MAX / 10 || (result == INT_MAX / 10 && digit > INT_MAX % 10)) {
            return (sign == 1) ? INT_MAX : INT_MIN;
        }

        result = result * 10 + digit;
        i++;
    }

    return result * sign;
}
int main() {
    printf("%d\n", myAtoi("42")); // output: 42
    printf("%d\n", myAtoi("   -42")); // output: -42
    printf("%d\n", myAtoi("4193 with words")); // output: 4193
    printf("%d\n", myAtoi("words and 987")); // output: 0
    printf("%d\n", myAtoi("-91283472332")); // output: -2147483648 (because the reversed value is out of range)
    return 0;
}
```

OUTPUT:

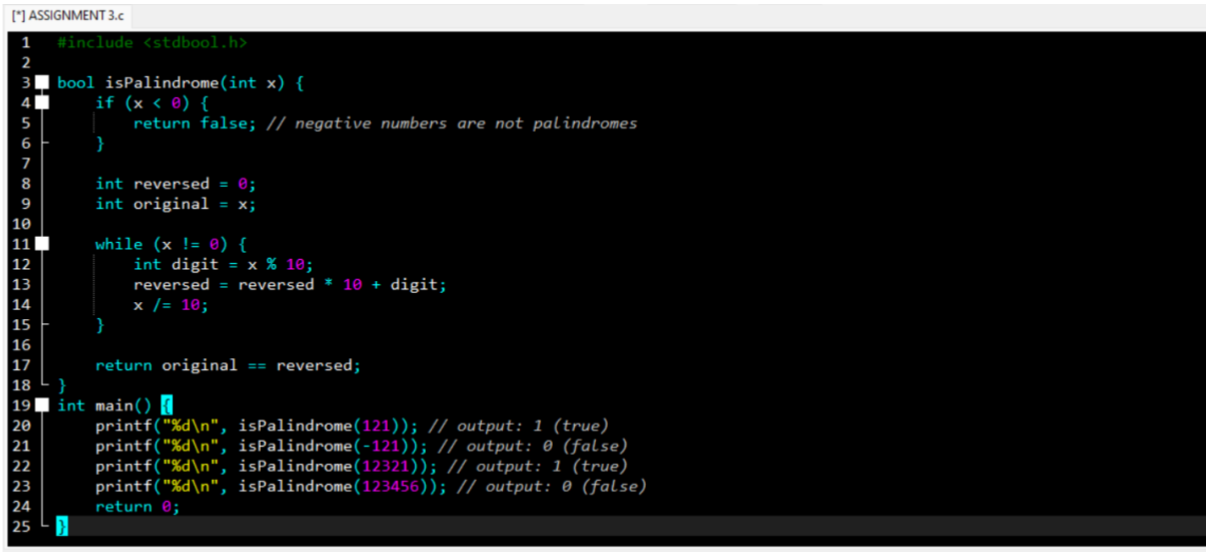


C:\Users\selco\OneDrive\Doc

```
42
-42
4193
0
-2147483648

--------------------------------
Process exited after 2.858 seconds with return value 0
Press any key to continue . . .
```

9)

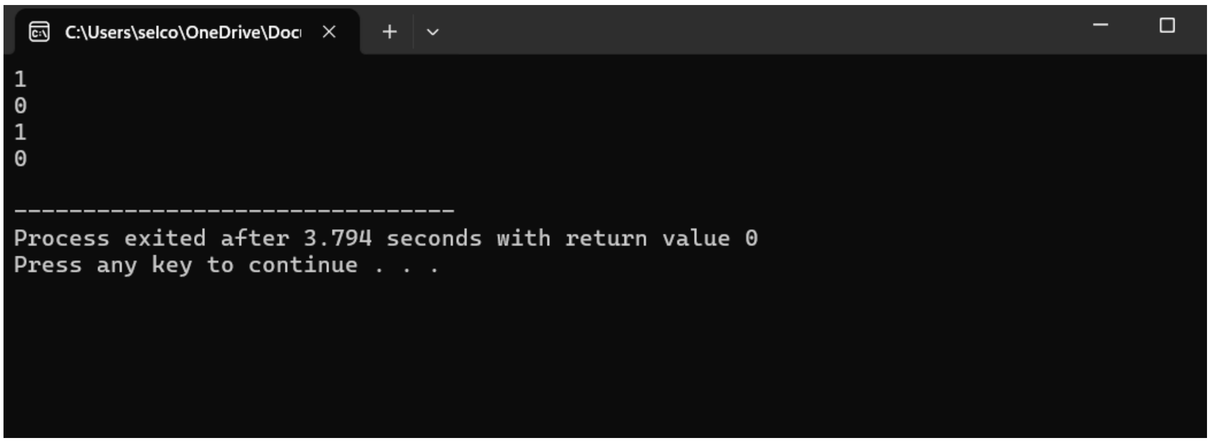
CODE:



[*] ASSIGNMENT 3.c

```c
#include <stdbool.h>

bool isPalindrome(int x) {
    if (x < 0) {
        return false; // negative numbers are not palindromes
    }

    int reversed = 0;
    int original = x;

    while (x != 0) {
        int digit = x % 10;
        reversed = reversed * 10 + digit;
        x /= 10;
    }

    return original == reversed;
}
int main() {
    printf("%d\n", isPalindrome(121)); // output: 1 (true)
    printf("%d\n", isPalindrome(-121)); // output: 0 (false)
    printf("%d\n", isPalindrome(12321)); // output: 1 (true)
    printf("%d\n", isPalindrome(123456)); // output: 0 (false)
    return 0;
}
```

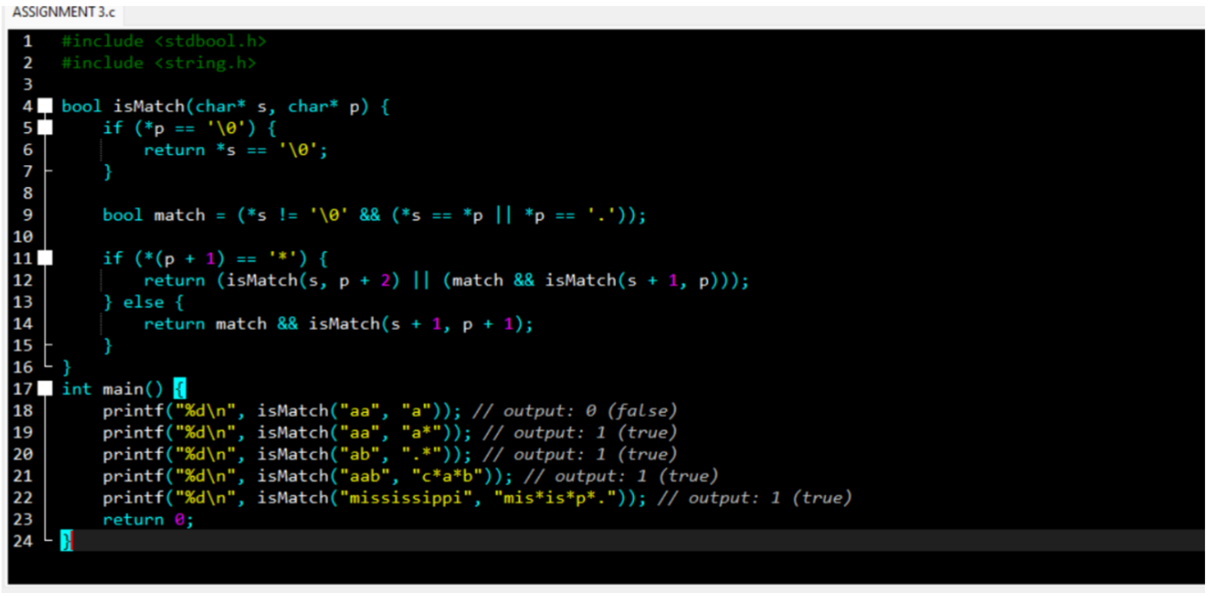
OUTPUT:



C:\Users\selco\OneDrive\Doc

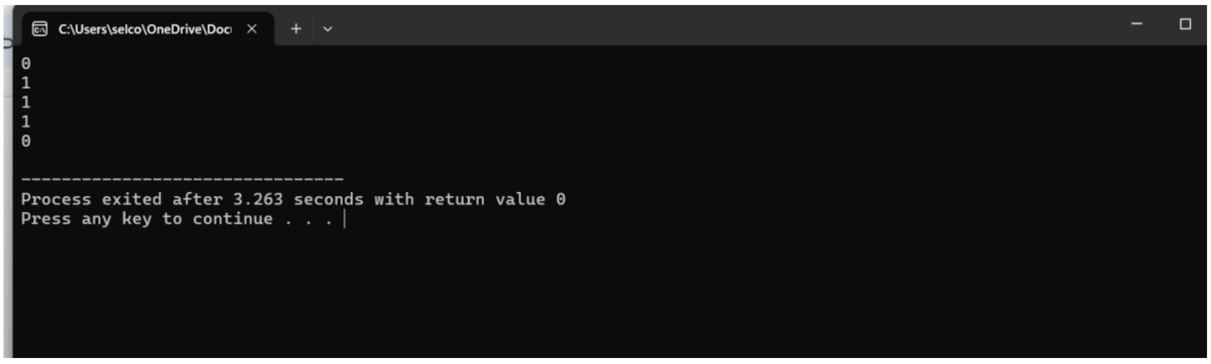
```
1
0
1
0

--------------------------------
Process exited after 3.794 seconds with return value 0
Press any key to continue . . .
```

10)
CODE:


ASSIGNMENT 3.c

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

bool isMatch(char* s, char* p) {
    if (*p == '\0') {
        return *s == '\0';
    }

    bool match = (*s != '\0' && (*s == *p || *p == '.'));

    if (*(p + 1) == '*') {
        return (isMatch(s, p + 2) || (match && isMatch(s + 1, p)));
    } else {
        return match && isMatch(s + 1, p + 1);
    }
}
int main() {
    printf("%d\n", isMatch("aa", "a")); // output: 0 (false)
    printf("%d\n", isMatch("aa", "a*")); // output: 1 (true)
    printf("%d\n", isMatch("ab", ".*")); // output: 1 (true)
    printf("%d\n", isMatch("aab", "c*a*b")); // output: 1 (true)
    printf("%d\n", isMatch("mississippi", "mis*is*p*.")); // output: 1 (true)
    return 0;
}
```

OUTPUT:


C:\Users\selco\OneDrive\Doc

```
0
1
1
1
0

--------------------------------
Process exited after 3.263 seconds with return value 0
Press any key to continue . . .
```