CSE-016 Programming Lab Assignment № 6

17/05/2024

Youssef Ahmed Samy Kassem
ID 9545 – Group 3 – Lab 1
SSP – Faculty of Engineering, Alexandria University

Solutions begin from the second page.
Screenshots of console input/ output come after all the source code.
Use Adobe Acrobat Reader to easily be able to copy the code.

1 Problems

1.1 Problem (1)

Write a program to compute the multiplication of matrices A and B. The program should read the values of A and B and print out the result. The dimensions of A and B are as follows: A[4][3] , B[3][2].

1.2 Problem (2)

Write a program to read a matrix of 4 rows and 3 columns, then it returns the value and location of the maximum element and the minimum element.

1.3 Problem (3)

Rewrite the Palindrome checking program (example 4) but use only one array for reading the input and then remove the spaces from it without using any additional arrays! [the size of the array should not exceed the length of input string]

Contents

2 Solutions - Source Code

2.1 Solution to Problem (1)

Program's C code | Line numbers to aid when copying

```c
#include <stdio.h>

int main()
{
    int i, j, k, A[4][3] = {0}, B[3][2] = {0}, R[4][2] = {0};
    printf("Instructions:\nInserting rows should be done in the format C1 C2 Cn\n");
    printf("=======Matrix A (4x3)=======\n");
    for (i = 0; i < 4; i++)
    {
        printf("[A] insert row (%d of 4) : 3 cols > ", (i+1));
        scanf("%d %d %d", &A[i][0], &A[i][1], &A[i][2]);
    }
    printf("=======Matrix B (3x2)=======\n");
    for (i = 0; i < 3; i++)
    {
        printf("[B] insert row (%d of 3) : 2 cols > ", (i+1));
        scanf("%d %d", &B[i][0], &B[i][1]);
    }
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 2; j++)
        {
            for (k = 0; k < 4; k++)
            {
                R[k][j] += A[k][i] * B[i][j];
            }
        }
    }
    printf("The result of the multiplication of the previous two matrices:");
    for (i = 0; i < 4; i++)
    {
        printf("\n[\t");
        for (j = 0; j < 2; j++)
        {
            printf("%d\t", R[i][j]);
        }
        printf("]");
    }
    return 0;
}
```

2.2 Solution to Problem (2)

Program's C code

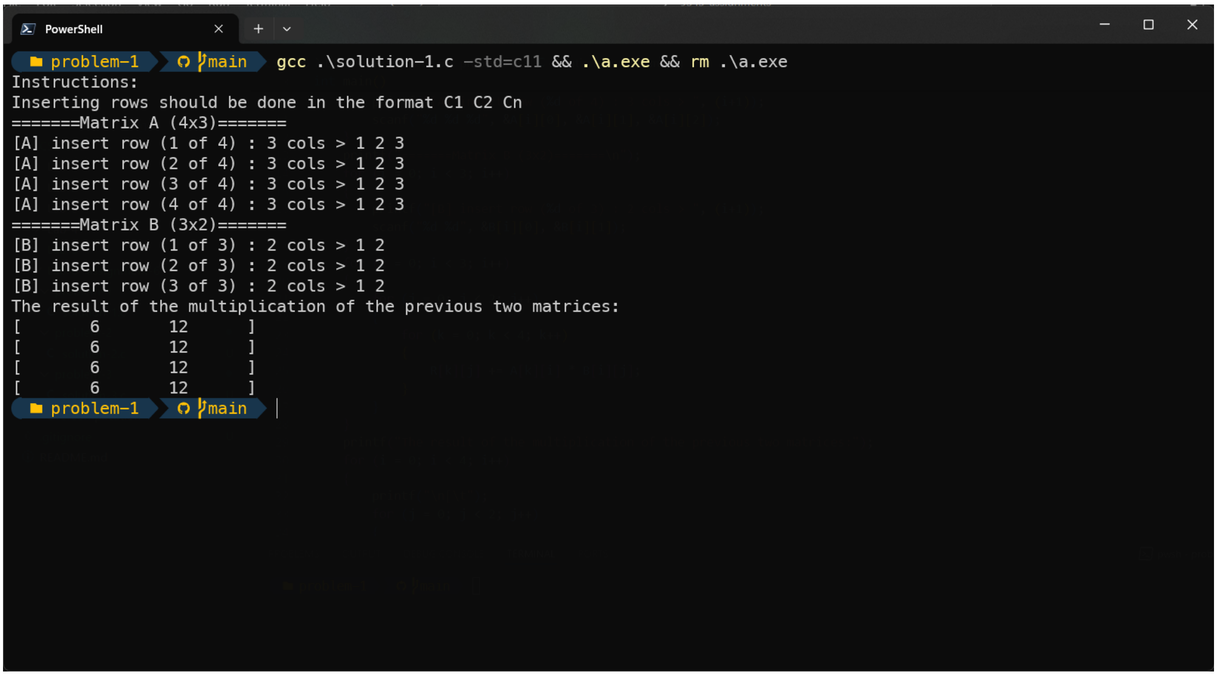
```c
#include <stdio.h>

int main()
{
    int i, j, M[4][3] = {0}, max, min, pos[2][2] = {{1, 1}, {1, 1}};
    printf("Instructions:\n");
    printf("You will provide a 4x3 matrix [M]\n");
    printf("You will enter 3 numbers per input (full row) in this format:\n");
    printf("C1 C2 C3\nWhere C stands for column.\n");
    printf("====Matrix [M]====\n");
    for (i = 0; i < 4; i++)
    {
        printf("[M] insert row (%d of 4) : 3 cols > ", (i + 1));
        scanf("%d %d %d", &M[i][0], &M[i][1], &M[i][2]);
    }
    max = min = M[0][0];
    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < 3; j++)
        {
            if (M[i][j] > max)
            {
                max = M[i][j];
                pos[0][0] = i + 1;
                pos[0][1] = j + 1;
            }
            else if (M[i][j] < min)
            {
                min = M[i][j];
                pos[1][0] = i + 1;
                pos[1][1] = j + 1;
            }
        }
    }
printf("Maximum number: %d. Position: Row %d, Column %d.\n",max,pos[0][0],pos[0][1]);
printf("Minimum number: %d. Position: Row %d, Column %d.\n",min,pos[1][0],pos[1][1]);
    return 0;
}
```

2.3 Solution to Problem (3)

Program's C code – Uses out-of-bounds indexes and unsafe function gets()

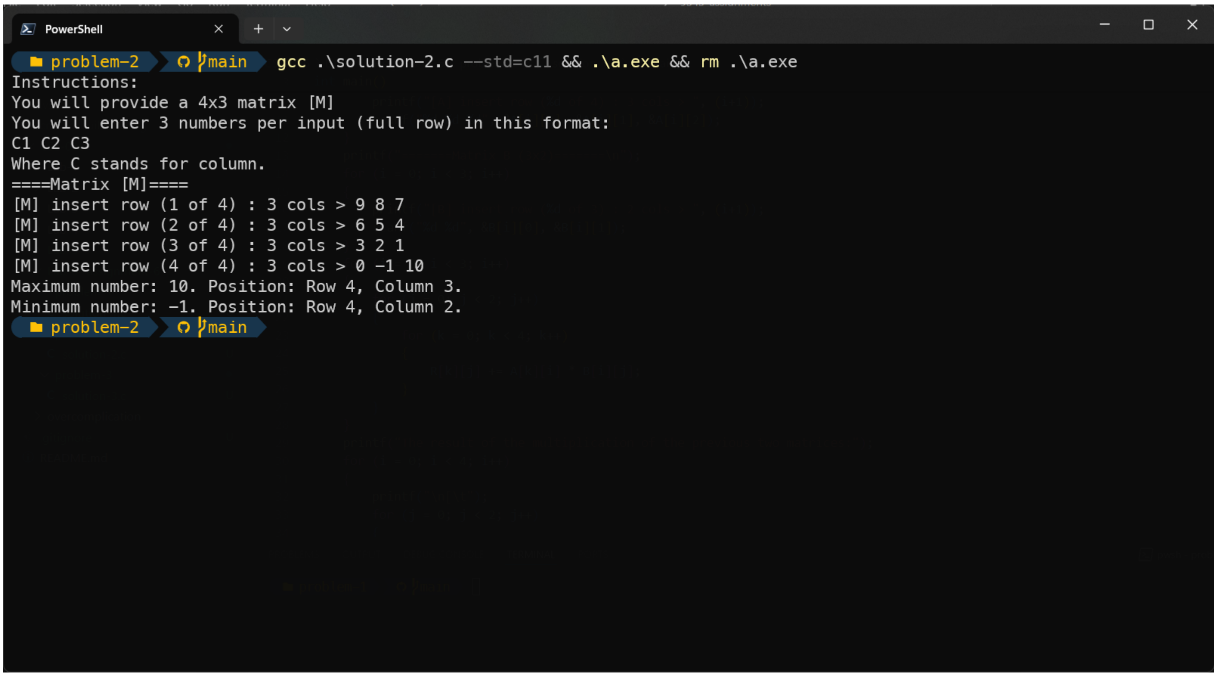
```c
#include <stdio.h>
#include <string.h>
int palindrome(char str[], int len);
char input[] = {};
int main()
{
    printf("Enter statement:\n");
    gets(input);
    int length = strlen(input), i, j;
    for (i = 0; i < length;)
    {
        if (input[i] == ' ')
        {
            for (j = i; j < length; j++)
            {
                input[j] = input[j + 1];
            }
            length--;
        }
        else
        {
            i++;
        }
    }
printf("%s : is %s palindrome.\n",input, (palindrome(input, length) ? "a" : "not a"));
    return 0;
}

int palindrome(char str[], int len)
{
    int i;
    len--;
    for (i = 0; i < len; i++)
    {
        if (str[i] != str[len - i])
            return 0;
    }
    return 1;
}
```

3 Console Input/ Output Screenshots



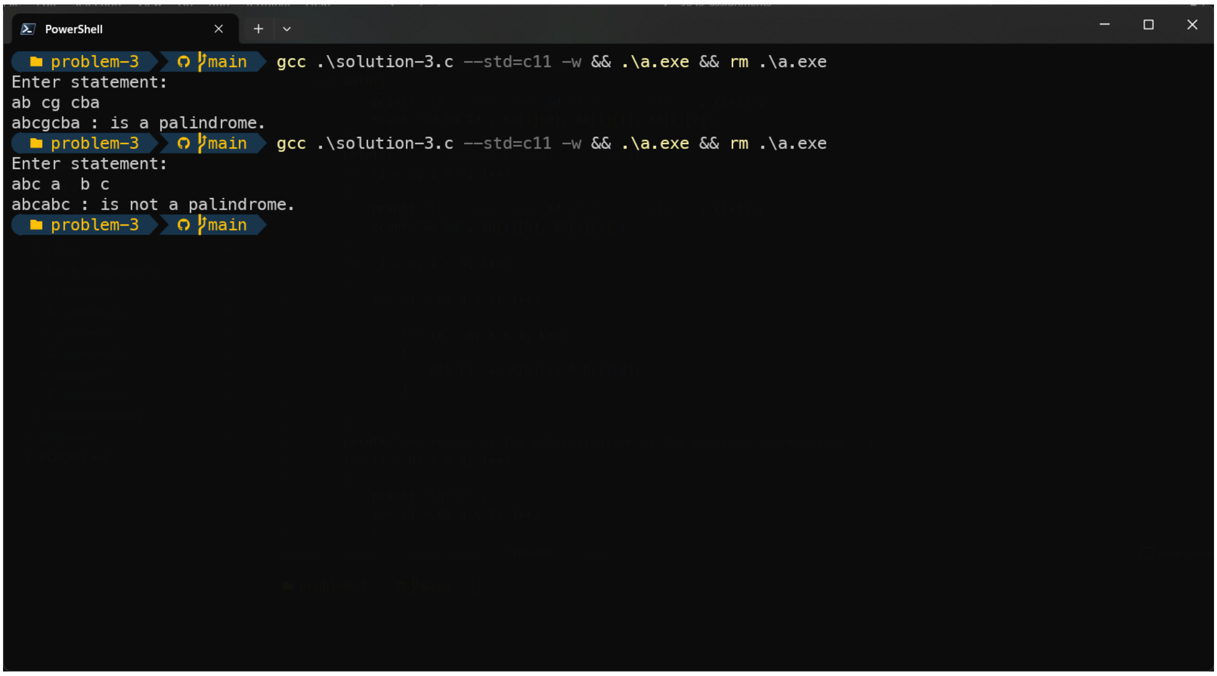
```
 problem-1   main  gcc .\solution-1.c -std=c11 && .\a.exe && rm .\a.exe
Instructions:
Inserting rows should be done in the format C1 C2 Cn
=======Matrix A (4x3)=======
[A] insert row (1 of 4) : 3 cols > 1 2 3
[A] insert row (2 of 4) : 3 cols > 1 2 3
[A] insert row (3 of 4) : 3 cols > 1 2 3
[A] insert row (4 of 4) : 3 cols > 1 2 3
=======Matrix B (3x2)=======
[B] insert row (1 of 3) : 2 cols > 1 2
[B] insert row (2 of 3) : 2 cols > 1 2
[B] insert row (3 of 3) : 2 cols > 1 2
The result of the multiplication of the previous two matrices:
[       6       12      ]
[       6       12      ]
[       6       12      ]
[       6       12      ]
 problem-1   main  |
```

Figure 1: Screenshot of Problem (1)'s program input/ output



```
 problem-2   main  gcc .\solution-2.c --std=c11 && .\a.exe && rm .\a.exe
Instructions:
You will provide a 4x3 matrix [M]
You will enter 3 numbers per input (full row) in this format:
C1 C2 C3
Where C stands for column.
====Matrix [M]====
[M] insert row (1 of 4) : 3 cols > 9 8 7
[M] insert row (2 of 4) : 3 cols > 6 5 4
[M] insert row (3 of 4) : 3 cols > 3 2 1
[M] insert row (4 of 4) : 3 cols > 0 -1 10
Maximum number: 10. Position: Row 4, Column 3.
Minimum number: -1. Position: Row 4, Column 2.
 problem-2   main 
```

Figure 2: Screenshot of Problem (2)'s program input/ output



```
 problem-3   main  gcc .\solution-3.c --std=c11 -w && .\a.exe && rm .\a.exe
Enter statement:
ab cg cba
abcgcba : is a palindrome.
 problem-3   main  gcc .\solution-3.c --std=c11 -w && .\a.exe && rm .\a.exe
Enter statement:
abc a  b c
abcabc : is not a palindrome.
 problem-3   main
```

Figure 3: Screenshot of Problem (3)'s program input/ output (Compiler warnings suppressed)

4 Specifications

- **Libraries:**
  - `stdio.h`
  - `string.h` (problem 3)
- **Compiler:** GNU C Compiler `(gcc)` version 13.2.0 (x86_64-posix-seh-rev1)
- **C Standard Compatibility**

P#	C89/C90	C99	C11	C17	C23
1	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓

- **Supported Platforms:** OS: (any), architecture: (any)
- **Tested On:** Windows 11 64-bit

5 Licenses

This document, my additions to its LaTeX source code, the software included and its `C` source code all come without warranty and are all subject to the BSD 3-Clause Open Source License: https://opensource.org/license/bsd-3-clause.