PATUAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

COURSE CODE CIT-112

SUBMITTED TO:

Md. Mahbubur Rahman

Department of Computer Science and Information Technology

Faculty of Computer Science and Engineering

SUBMITTED BY:

Md. Sharafat Karim

ID: **2102024,**

Registration No: **10151**

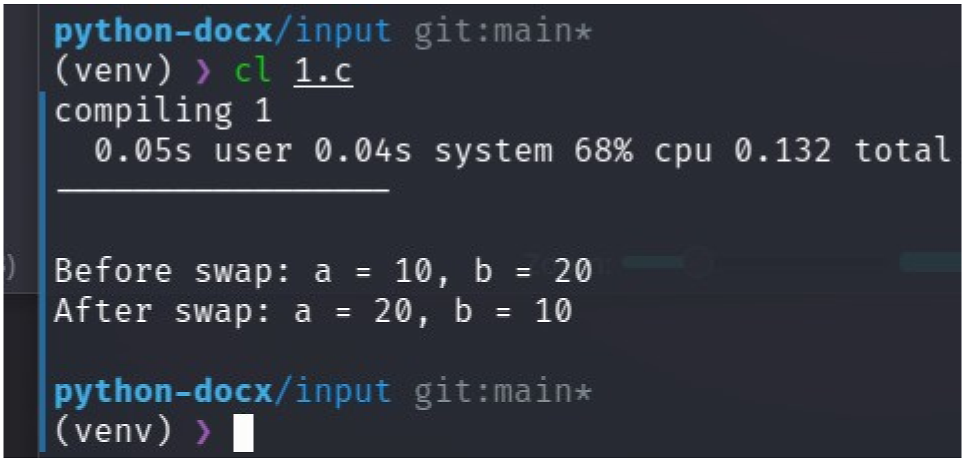
Faculty of Computer Science and Engineering

Assignment: **08**

Table of Contents

1 Write a function exchange to interchange the values of two variables, say x and y. Illustrate the use of this function, in a calling function. Assume that x and y are defined as global variables.

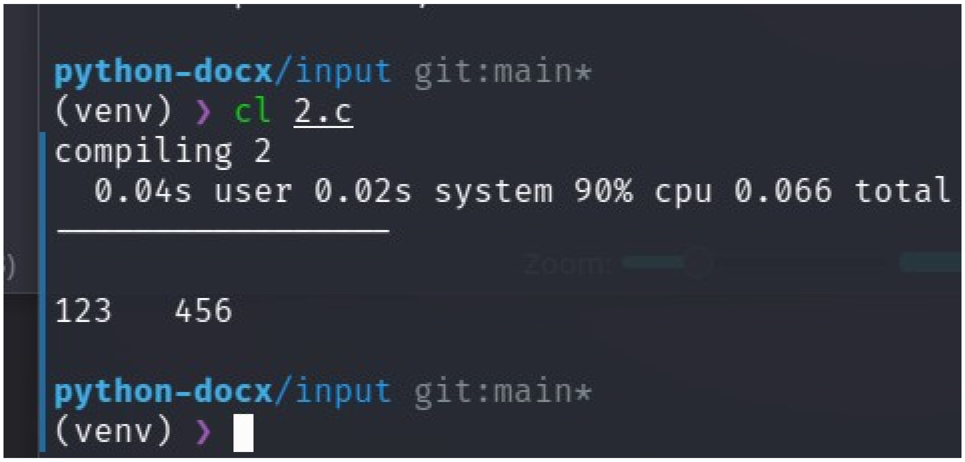
```
#include<stdio.h>

int a = 10, b = 20;

void swap (void)
{
   b = a + b ;
   a = b - a ;
   b = b - a ;
}

int main()
{
   printf("Before swap: a = %d, b = %d\n", a, b);
   swap();
   printf("After swap: a = %d, b = %d\n", a, b);
   return 0;
}
```



```
python-docx/input git:main*
(venv) ❯ cl 1.c
compiling 1
  0.05s user 0.04s system 68% cpu 0.132 total
_________________

Before swap: a = 10, b = 20
After swap: a = 20, b = 10

python-docx/input git:main*
(venv) ❯ 
```

2 Write a function space(x) that can be used to provide a space of x positions between two output numbers. Demonstrate its application.

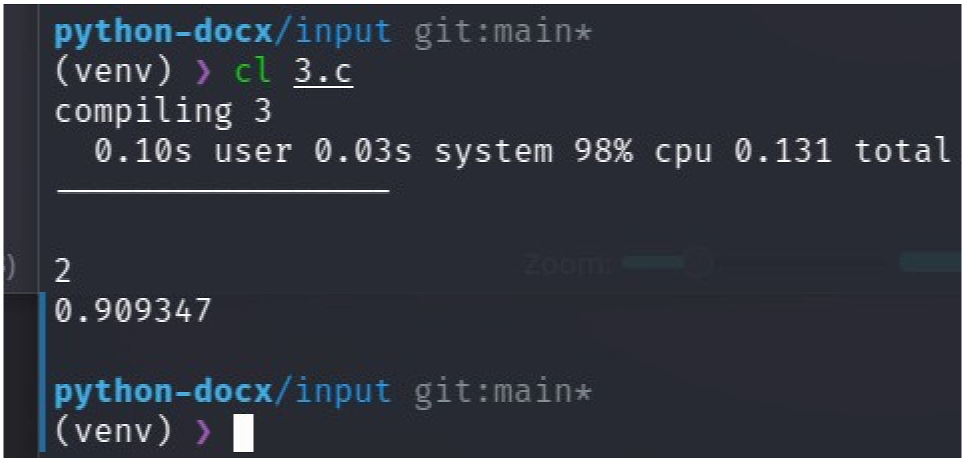
```
#include <stdio.h>

void space(int x) {
  for (int i = 0; i < x; i++) {
    printf(" ");
  }
}

int main() {
  printf("123");
  space(3);
  printf("456\n");
  return 0;
}
```



```
python-docx/input git:main*
(venv) ❯ cl 2.c
compiling 2
  0.04s user 0.02s system 90% cpu 0.066 total
_________________

123   456

python-docx/input git:main*
(venv) ❯ █
```

3 Use recursive function calls to evaluate

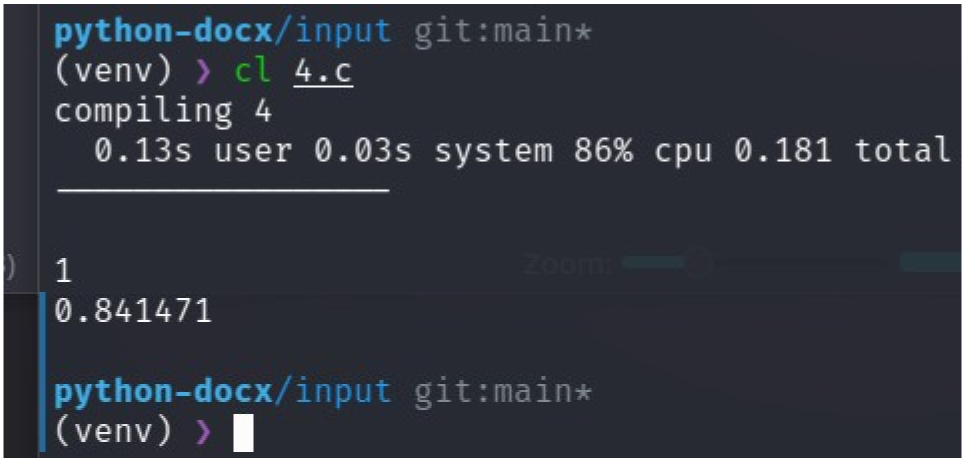
```c
#include <stdio.h>
#include <math.h>

int factorial(int n)
{
    if (n == 1)
        return 1;
    else
        return n * factorial(n - 1);
}

float evaluate(int x, int n, int i)
{
    if (n >= 10)
        return 0;
    else if (i % 2 == 0)
        return -pow(x, n) / factorial(n) + evaluate(x, n + 2, i+1);
    else
        return pow(x, n) / factorial(n) + evaluate(x, n + 2, i+1);
}

int main()
{
    int n;
    scanf("%d", &n);
    printf("%f\n", evaluate(n, 1, 1));
}
```



```
python-docx/input git:main*
(venv) ❯ cl 3.c
compiling 3
  0.10s user 0.03s system 98% cpu 0.131 total
_________________


2
0.909347

python-docx/input git:main*
(venv) ❯ █
```

4 Write a function to evaluate the polynomial, using an array variable.

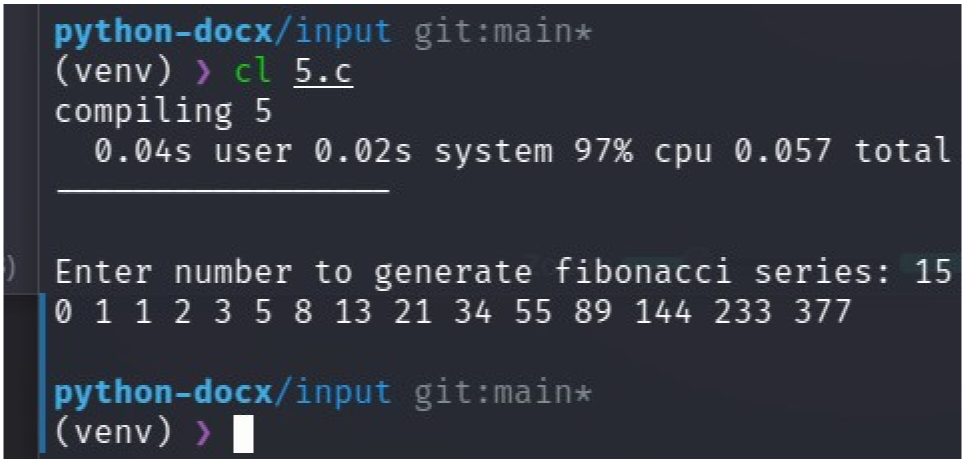
```
// n order polinoial
// Generated with AI

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

int factorial(int n)
{
    if (n == 1)
        return 1;
    else
        return n * factorial(n - 1);
}

float evaluate(int x, int n, int i)
{
    if (n >= 11)
        return 0;
    else if (i % 2 == 0)
        return -pow(x, n) / factorial(n) + evaluate(x, n + 2, i+1);
    else
        return pow(x, n) / factorial(n) + evaluate(x, n + 2, i+1);
}

int main()
{
    int n;
    scanf("%d", &n);
    printf("%f\n", evaluate(n, 1, 1));
}
```



```
python-docx/input git:main*
(venv) ❯ cl 4.c
compiling 4
  0.13s user 0.03s system 86% cpu 0.181 total
─────────────────


1
0.841471

python-docx/input git:main*
(venv) ❯ 
```

5 Write a function that will generate and print the first n Fibonacci numbers. Test the function for n = 5, 10, and 15.

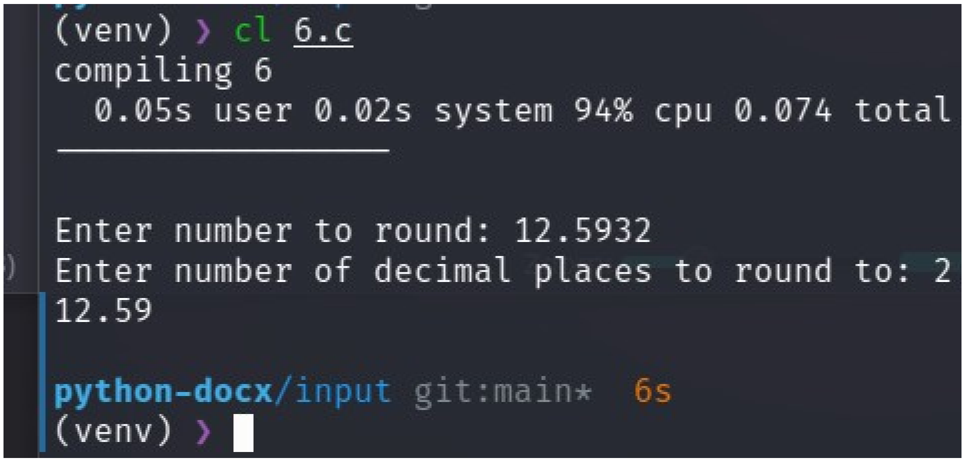
```
#include <stdio.h>

void fibonacci(int n)
{
    int i, a = 0, b = 1, c;
    for (i = 0; i < n; i++)
    {
        printf("%d ", a);
        c = a + b;
        a = b;
        b = c;
    }
}

int main()
{
    int n;
    printf("Enter number to generate fibonacci series: ");
    scanf("%d", &n);
    fibonacci(n);
    printf("\n");
}
```



```
python-docx/input git:main*
(venv) ❯ cl 5.c
compiling 5
  0.04s user 0.02s system 97% cpu 0.057 total
─────────────────

Enter number to generate fibonacci series: 15
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

python-docx/input git:main*
(venv) ❯ 
```

6 Write a function that will round a floating-point number to an indicated decimal place. For example the number 17.457 would yield the value 17.46 when it is rounded off to two decimal places.

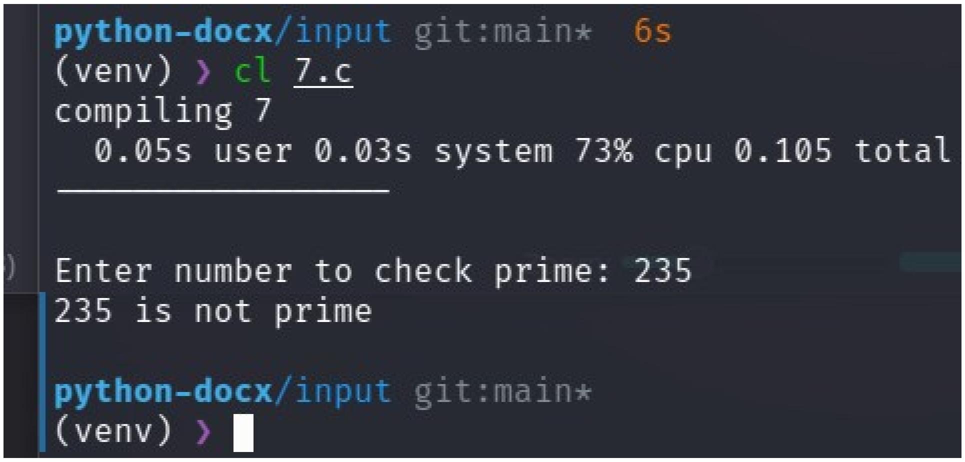
```
#include <stdio.h>

void rounded(float num, int round)
{
   printf("%.*f\n", round, num);
}

int main()
{
   float num;
   int round;

   printf("Enter number to round: ");
   scanf("%f", &num);
   printf("Enter number of decimal places to round to: ");
   scanf("%d", &round);

   rounded(num, round);
}
```



```
(venv) ❯ cl 6.c
compiling 6
  0.05s user 0.02s system 94% cpu 0.074 total
________________


Enter number to round: 12.5932
Enter number of decimal places to round to: 2
12.59

python-docx/input git:main*  6s
(venv) ❯ █
```

7 Write a function prime that returns 1 if its argument is a prime number and returns zero otherwise.

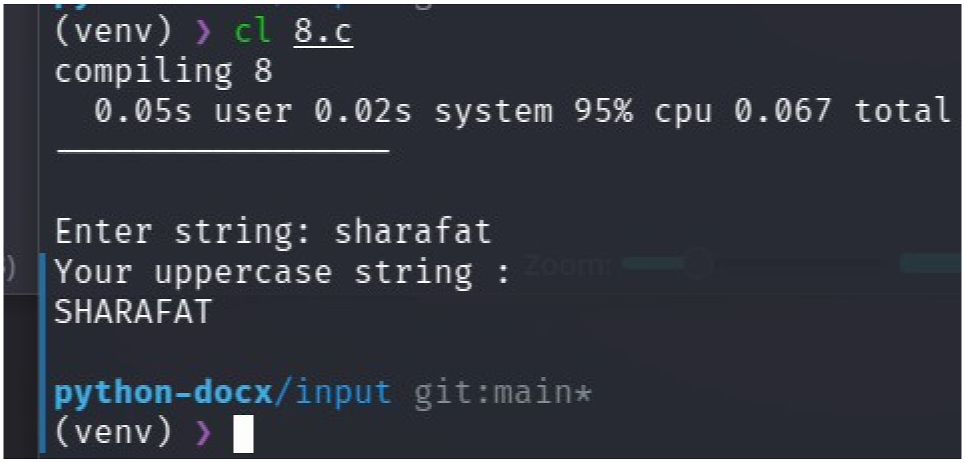
```
#include <stdio.h>

int check_prime(int number_to_check_prime)
{
    int i;
    for (i = 2; i < number_to_check_prime; i++)
    {
        if (number_to_check_prime % i == 0)
            return 0;
    }
    return 1;
}

int main()
{
    int number_to_check_prime;
    printf("Enter number to check prime: ");
    scanf("%d", &number_to_check_prime);

    if (check_prime(number_to_check_prime))
        printf("%d is prime\n", number_to_check_prime);
    else
        printf("%d is not prime\n", number_to_check_prime);
}
```



```
python-docx/input git:main*   6s
(venv) ❯ cl 7.c
compiling 7
  0.05s user 0.03s system 73% cpu 0.105 total
_________________

Enter number to check prime: 235
235 is not prime

python-docx/input git:main*
(venv) ❯
```

8 Write a function that will scan a character string passed as an argument and convert all lowercase characters into their uppercase equivalents.

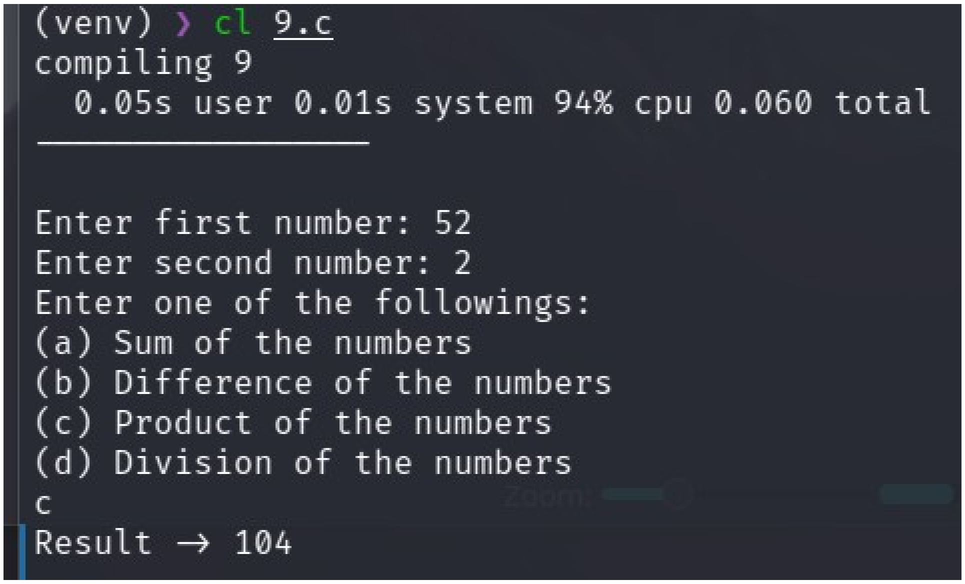
```
#include <stdio.h>

void to_uppercase(char *string)
{
    int i;
    for (i = 0; string[i] != '\0'; i++)
    {
        if (string[i] >= 'a' && string[i] <= 'z')
            string[i] -= 32;
    }
}

int main()
{
    char string[100];
    printf("Enter string: ");
    scanf("%s", string);

    to_uppercase(string);
    printf("Your uppercase string : \n");
    printf("%s\n", string);
    return 0;
}
```



```
(venv) ❯ cl 8.c
compiling 8
  0.05s user 0.02s system 95% cpu 0.067 total
_________________

Enter string: sharafat
Your uppercase string :
SHARAFAT

python-docx/input git:main*
(venv) ❯
```

9 Develop a top_down modular program to implement a calculator. The program should request the user to input two numbers and display one of the following as per the desire of the user:

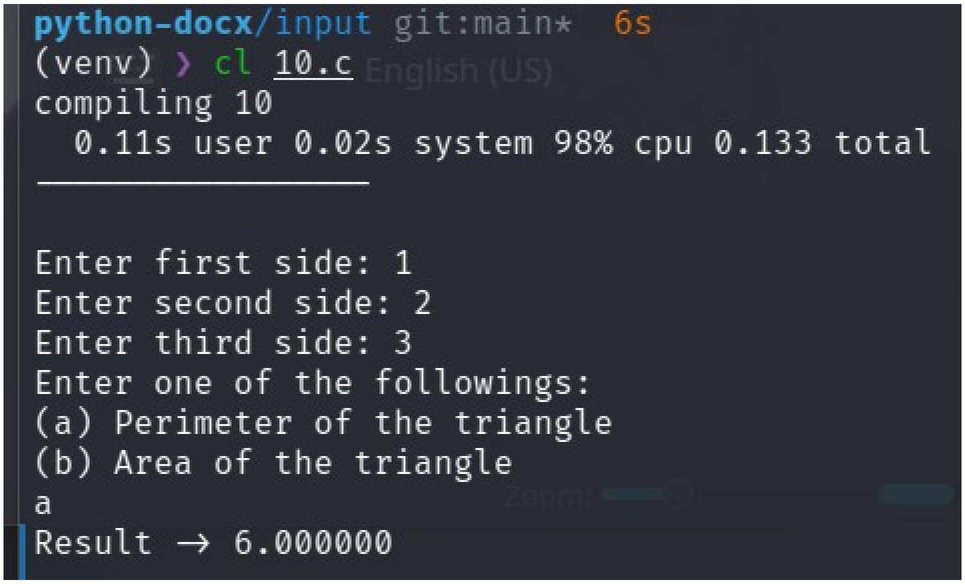
```
#include <stdio.h>

int sum(int a, int b)
{
    return a + b;
}

int difference(int a, int b)
{
    return a - b;
}

int product(int a, int b)
{
    return a * b;
}

int division(int a, int b)
{
    return a / b;
}

int main()
{
    int a, b;
    char operation;

    printf("Enter first number: ");
    scanf("%d", &a);
    printf("Enter second number: ");
    scanf("%d", &b);
    printf("Enter one of the followings: ");
    printf("\n(a) Sum of the numbers ");
    printf("\n(b) Difference of the numbers ");
    printf("\n(c) Product of the numbers ");
    printf("\n(d) Division of the numbers ");
    printf("\n");

    scanf(" %c", &operation);
    switch (operation)
    {
        case 'a':
            printf("Result -> %d\n", sum(a, b));
            break;
        case 'b':
            printf("Result -> %d\n", difference(a, b));
            break;
        case 'c':
            printf("Result -> %d\n", product(a, b));
            break;
        case 'd':
            printf("Result -> %d\n", division(a, b));
            break;
        default:
            printf("Invalid operation\n");
    }
}
```



```
(venv) ❯ cl 9.c
compiling 9
  0.05s user 0.01s system 94% cpu 0.060 total


Enter first number: 52
Enter second number: 2
Enter one of the followings:
(a) Sum of the numbers
(b) Difference of the numbers
(c) Product of the numbers
(d) Division of the numbers
c
Result -> 104
```

10 Develop a modular interactive program using functions that reads the values of three sides of a triangle and displays either its area or its perimeter as per the request of the user. Given the three sides a, b and c.

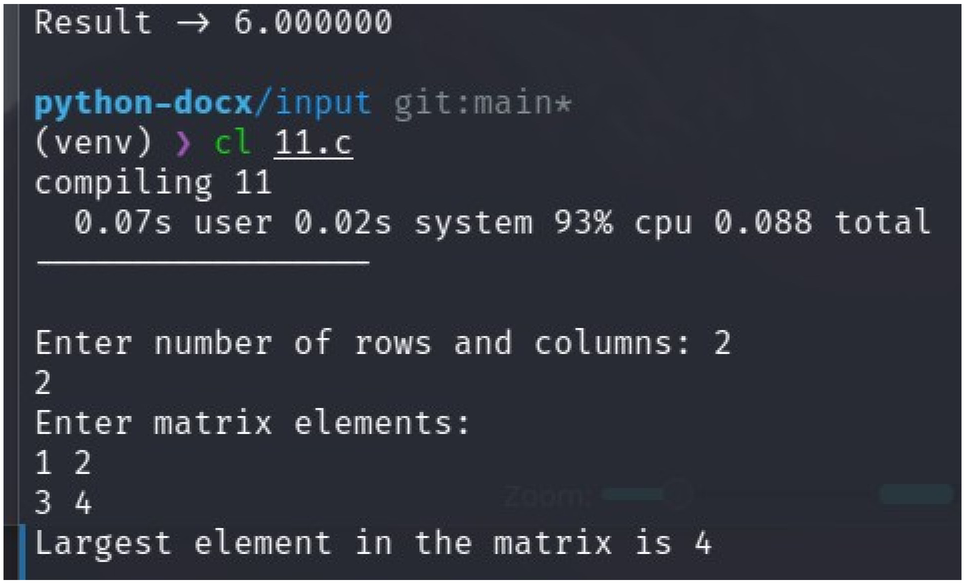
```
#include <stdio.h>
#include <math.h>

float perimeter(float a, float b, float c)
{
    return a + b + c;
}

float area(float a, float b, float c)
{
    float s = (a + b + c) / 2;
    return sqrt((s - a) * (s - b) * (s - c));
}

int main()
{
    float a, b, c;
    char operation;

    printf("Enter first side: ");
    scanf("%f", &a);
    printf("Enter second side: ");
    scanf("%f", &b);
    printf("Enter third side: ");
    scanf("%f", &c);

    printf("Enter one of the followings: ");
    printf("\n(a) Perimeter of the triangle ");
    printf("\n(b) Area of the triangle ");
    printf("\n");
    scanf(" %c", &operation);
    switch (operation)
    {
        case 'a':
            printf("Result -> %f\n", perimeter(a, b, c));
            break;
        case 'b':
            printf("Result -> %f\n", area(a, b, c));
            break;
        default:
            printf("Invalid operation\n");
    }
}
```



```
python-docx/input git:main*  6s
(venv) ❯ cl 10.c
compiling 10
  0.11s user 0.02s system 98% cpu 0.133 total
─────────────────

Enter first side: 1
Enter second side: 2
Enter third side: 3
Enter one of the followings:
(a) Perimeter of the triangle
(b) Area of the triangle
a
Result -> 6.000000
```

11 Write a function that can be called to find the largest element of an m by n matrix.

```
#include <stdio.h>

int find_largest(int *matrix, int rows, int cols) {
    int i, j, largest = *matrix;
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            if (*(matrix + i * cols + j) > largest) {
                largest = *(matrix + i * cols + j);
            }
        }
    }
    return largest;
}

int main() {
    int m, n, i, j;
    printf("Enter number of rows and columns: ");
    scanf("%d %d", &m, &n);

    int matrix[m][n];
    printf("Enter matrix elements: \n");
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }

    int largest = find_largest(&matrix[0][0], m, n);
    printf("Largest element in the matrix is %d\n", largest);

    return 0;
}
```

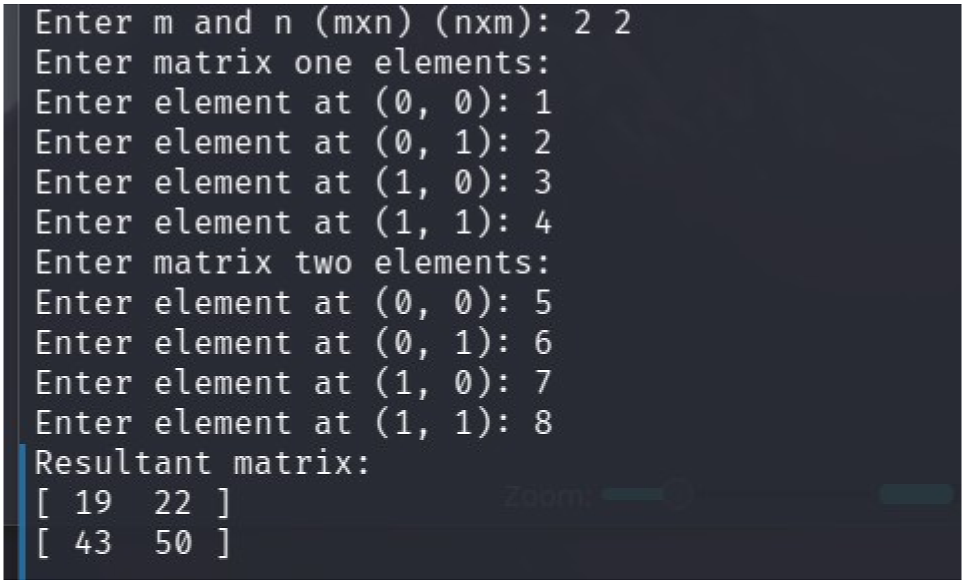


```
Result → 6.000000

python-docx/input git:main*
(venv) ❯ cl 11.c
compiling 11
  0.07s user 0.02s system 93% cpu 0.088 total
─────────────────


Enter number of rows and columns: 2
2
Enter matrix elements:
1 2
3 4
Largest element in the matrix is 4
```

12 Write a function that can be called to compute the product of two matrices of size m by n and n by m. The main function provides the values for m and n and two matrices.

```
#include <stdio.h>

void multiply_matrices(int *matrix_one, int *matrix_two, int m, int n)
{
    int i, j, k;
    int result[m][m];

    for (i = 0; i < m; i++)
    {
        for (j = 0; j < m; j++)
        {

            result[i][j] = 0;
            for (k = 0; k < n; k++)
            {
                result[i][j] += *(matrix_one + i * n + k) * *(matrix_two + k * m + j);
            }
        }
    }

    printf("Resultant matrix: \n");
    for (i = 0; i < m; i++)
    {
        printf("[");
        for (j = 0; j < m; j++)
        {
            printf(" %d ", result[i][j]);
        }
        printf("]\n");
    }
}
```

```
int main()
{
    int m, n;
    printf("Enter m and n (mxn) (nxm): ");
    scanf("%d %d", &m, &n);

    int matrix_one[m][n];
    int matrix_two[n][m];
    int i, j;

    printf("Enter matrix one elements: \n");
    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
        {
            printf("Enter element at (%d, %d): ", i, j);
            scanf("%d", &matrix_one[i][j]);
        }
    }

    printf("Enter matrix two elements: \n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < m; j++)
        {
            printf("Enter element at (%d, %d): ", i, j);
            scanf("%d", &matrix_two[i][j]);
        }
    }

    multiply_matrices((int *)matrix_one, (int *)matrix_two, m, n);
}
```

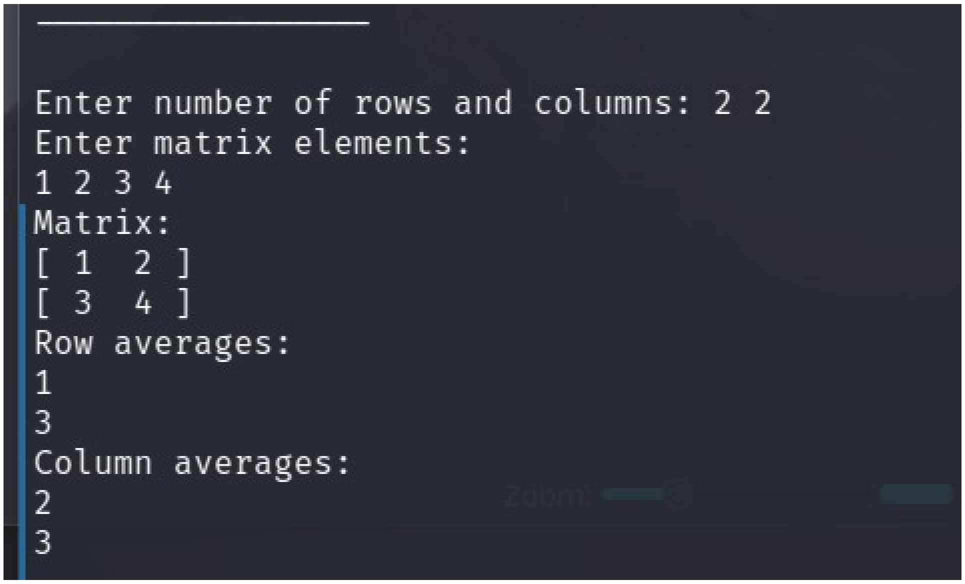


```
Enter m and n (mxn) (nxm): 2 2
Enter matrix one elements:
Enter element at (0, 0): 1
Enter element at (0, 1): 2
Enter element at (1, 0): 3
Enter element at (1, 1): 4
Enter matrix two elements:
Enter element at (0, 0): 5
Enter element at (0, 1): 6
Enter element at (1, 0): 7
Enter element at (1, 1): 8
Resultant matrix:
[ 19  22 ]
[ 43  50 ]
```

13 Design and code an interactive modular program that will use functions to a matrix of m by n size, compute column averages and row averages, and then print the entire matrix with averages shown in respective rows and columns.

```
#include <stdio.h>

void input_matrix(int *matrix, int rows, int cols)
{
    int i, j;
    printf("Enter matrix elements: \n");
    for (i = 0; i < rows; i++)
    {
        for (j = 0; j < cols; j++)
            scanf("%d", &matrix[i * cols + j]);
    }
}

void print_matrix(int *matrix, int rows, int cols)
{
    int i, j;
    printf("Matrix: \n");
    for (i = 0; i < rows; i++)
    {
        printf("[");
        for (j = 0; j < cols; j++)
            printf(" %d ", matrix[i * cols + j]);
        printf("]\n");
    }
}

void print_row_averages(int *matrix, int rows, int cols)
{
    int i, j;
    printf("Row averages: \n");
    for (i = 0; i < rows; i++)
    {
        int sum = 0;
        for (j = 0; j < cols; j++)
            sum += matrix[i * cols + j];
        printf("%d\n", sum / cols);
    }
}

void print_col_averages(int *matrix, int rows, int cols)
{
    int i, j;
    printf("Column averages: \n");
    for (i = 0; i < cols; i++)
    {
        int sum = 0;
        for (j = 0; j < rows; j++)
            sum += matrix[j * cols + i];
        printf("%d\n", sum / rows);
    }
}

int main()
{
    int m, n;
    printf("Enter number of rows and columns: ");
    scanf("%d %d", &m, &n);

    int matrix[m][n];
    input_matrix(&matrix[0][0], m, n);

    print_matrix(&matrix[0][0], m, n);
    print_row_averages(&matrix[0][0], m, n);
    print_col_averages(&matrix[0][0], m, n);

    return 0;
```

```
}
```

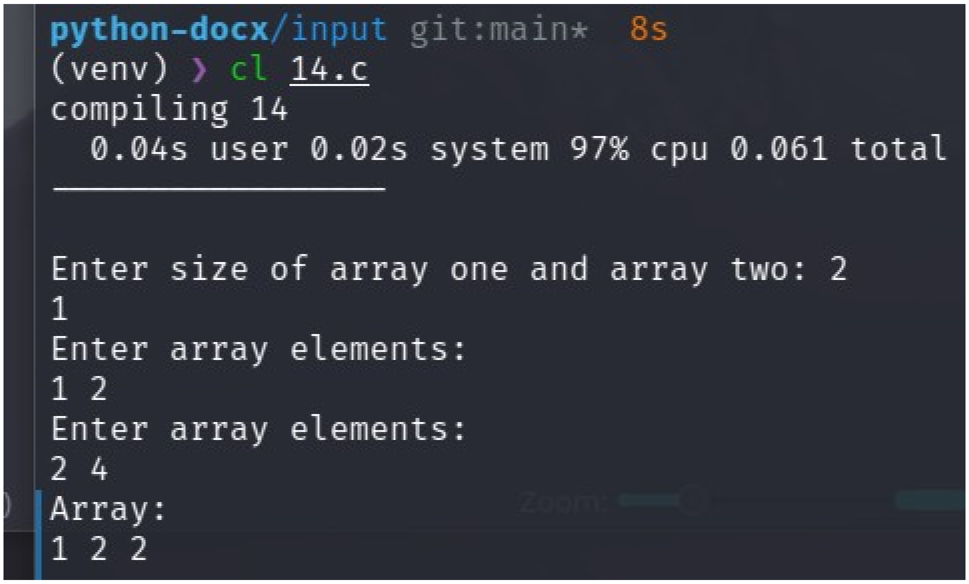


```
Enter number of rows and columns: 2 2
Enter matrix elements:
1 2 3 4
Matrix:
[ 1  2 ]
[ 3  4 ]
Row averages:
1
3
Column averages:
2
3
```

14 modular program of array

```
// Develop a top-down modular program that will perform the following tasks:
// ```
// (a) Read two integer arrays with unsorted elements.
// (b) Sort them in ascending order
// (c) Merge the sorted arrays
// (d) Print the sorted list
// ```
// Use functions for carrying out each of the above tasks. The main function should have only function calls.

#include <stdio.h>

void input_array(int *array, int size)
{
    int i;
    printf("Enter array elements: \n");
    for (i = 0; i < size; i++)
        scanf("%d", &array[i]);
}

void print_array(int *array, int size)
{
    int i;
    printf("Array: \n");
    for (i = 0; i < size; i++)
        printf("%d ", array[i]);
    printf("\n");
}

void sort_array(int *array, int size)
{
    int i, j;
    for (i = 0; i < size; i++)
```

```
    {
        int min = array[i], min_index = i;
        for (j = i + 1; j < size; j++)
        {
            if (array[j] < min)
            {
                min = array[j];
                min_index = j;
            }
        }
        int temp = array[i];
        array[i] = array[min_index];
        array[min_index] = temp;
    }
}

void merge_arrays(int *array_one, int *array_two, int *result, int size_one, int
size_two)
{
    int i, j;
    for (i = 0; i < size_one; i++)
        result[i] = array_one[i];
    for (j = 0; j < size_two; j++)
        result[i + j] = array_two[j];
}

void operations(void)
{
    int m, n;
    printf("Enter size of array one and array two: ");
    scanf("%d %d", &m, &n);
    int array_one[m], array_two[n];
    input_array(array_one, m);
    input_array(array_two, n);

    sort_array(array_one, m);
    sort_array(array_two, n);
    int result[m + n];
    merge_arrays(array_one, array_two, result, m, n);
    sort_array(result, m+n);

    print_array(result, m + n);
}

int main()
{
    operations();
    return 0;
}
```

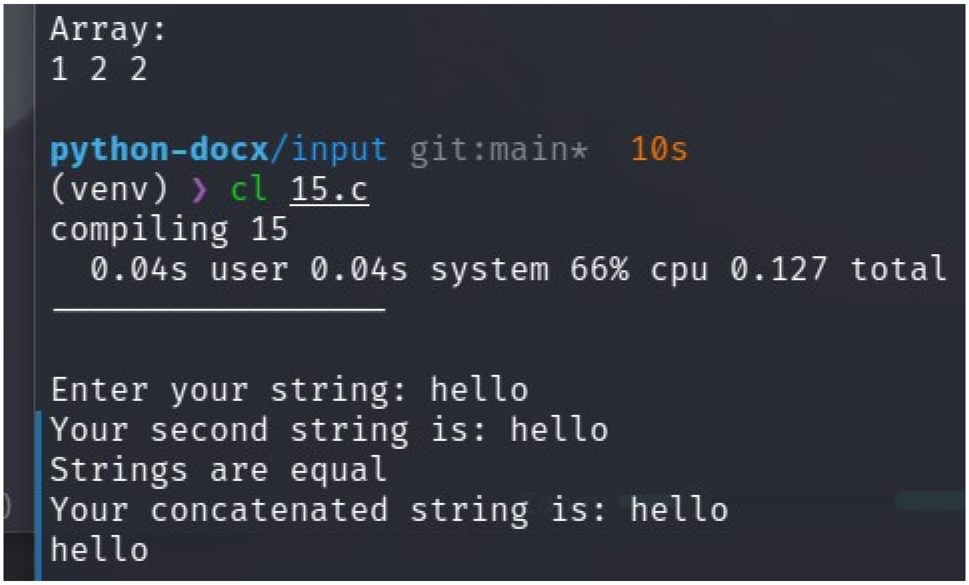


```
python-docx/input git:main*  8s
(venv) ❯ cl 14.c
compiling 14
  0.04s user 0.02s system 97% cpu 0.061 total
_________________

Enter size of array one and array two: 2
1
Enter array elements:
1 2
Enter array elements:
2 4
Array:
1 2 2
```

15 string operation

```
#include <stdio.h>

void copy_string(char *one, char *two)
{
    int i;
    for (i=0; one[i] != '\0'; i++)
        two[i] = one[i];
    two[i] = '\0';
    return;
}

void compare_string(char *one, char *two)
{
    int i;
    for (i=0; one[i] != '\0'; i++)
        if (one[i] != two[i])
            break;
    if (one[i] == '\0' && two[i] == '\0')
        printf("Strings are equal\n");
    else
        printf("Strings are not equal\n");
    return;
}

void concat_string(char *one, char *two)
{
    int i, j;
    for (i=0; one[i] != '\0'; i++);
    for (j=0; two[j] != '\0'; j++)
        one[i+j] = two[j];
    one[i+j] = '\0';
    return;
}
```

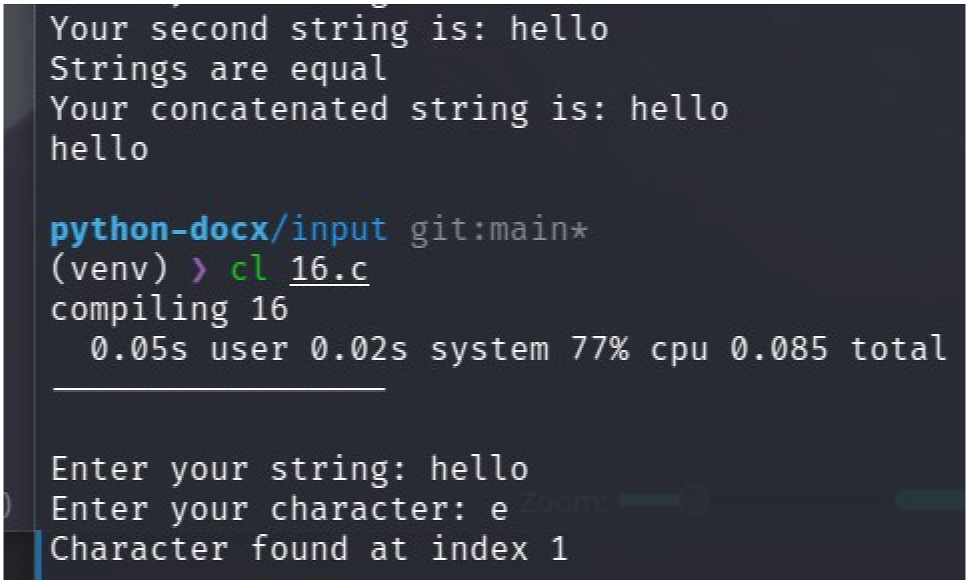
```
int main()
{
    char string_one[100], string_two[200];
    printf("Enter your string: ");
    fgets(string_one, 100, stdin);

    copy_string(string_one, string_two);
    printf("Your second string is: %s", string_two);

    compare_string(string_one, string_two);

    concat_string(string_one, string_two);
    printf("Your concatenated string is: %s", string_one);

    return 0;
}
```



```
Array:
1 2 2

python-docx/input git:main*  10s
(venv) ❯ cl 15.c
compiling 15
  0.04s user 0.04s system 66% cpu 0.127 total
_________________

Enter your string: hello
Your second string is: hello
Strings are equal
Your concatenated string is: hello
hello
```

16 Write a program that invokes a function called find( ) to perform the following tasks:

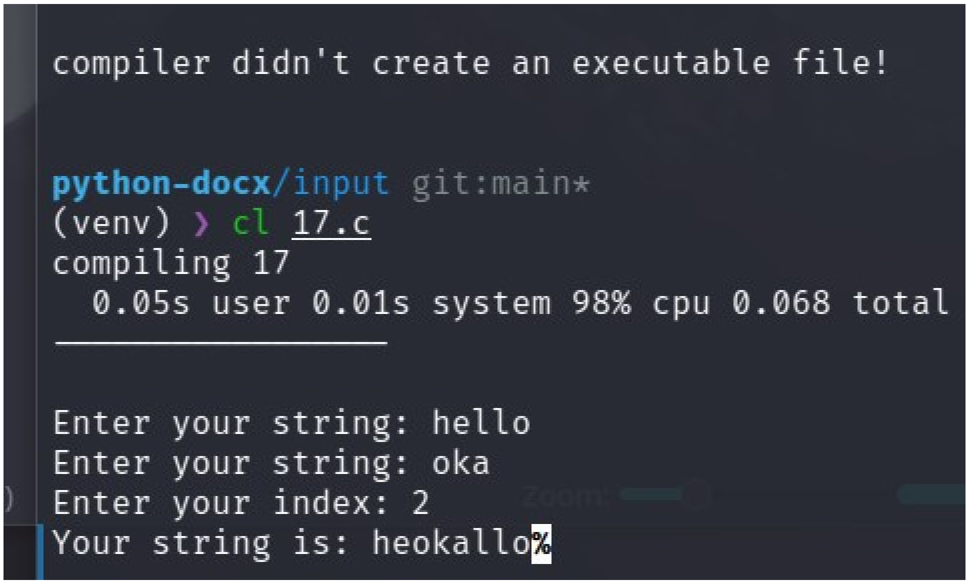
```
#include <stdio.h>

int char_search_inside_string(char *string, char c)
{
    int i;
    for (i=0; string[i] != '\0'; i++)
        if (string[i] == c)
            return i;
    return -1;
}

int main()
{
    char string[100], c;
    printf("Enter your string: ");
    fgets(string, 100, stdin);
    printf("Enter your character: ");
    scanf("%c", &c);

    int index = char_search_inside_string(string, c);
    if (index == -1)
        printf("Character not found\n");
    else
        printf("Character found at index %d\n", index);


return
0;
}
```



```
Your second string is: hello
Strings are equal
Your concatenated string is: hello
hello

python-docx/input git:main*
(venv) ❯ cl 16.c
compiling 16
  0.05s user 0.02s system 77% cpu 0.085 total
 ________________

Enter your string: hello
Enter your character: e
Character found at index 1
```

17 Design a function locate ( ) that takes two character arrays s1 and s2 and one integer value m as parameters and inserts the string s2 into s1 immediately after the index m . Write a program to test the function using a real-life situation. (Hint: s2 may be a missing word in s1 that represents a line of text).

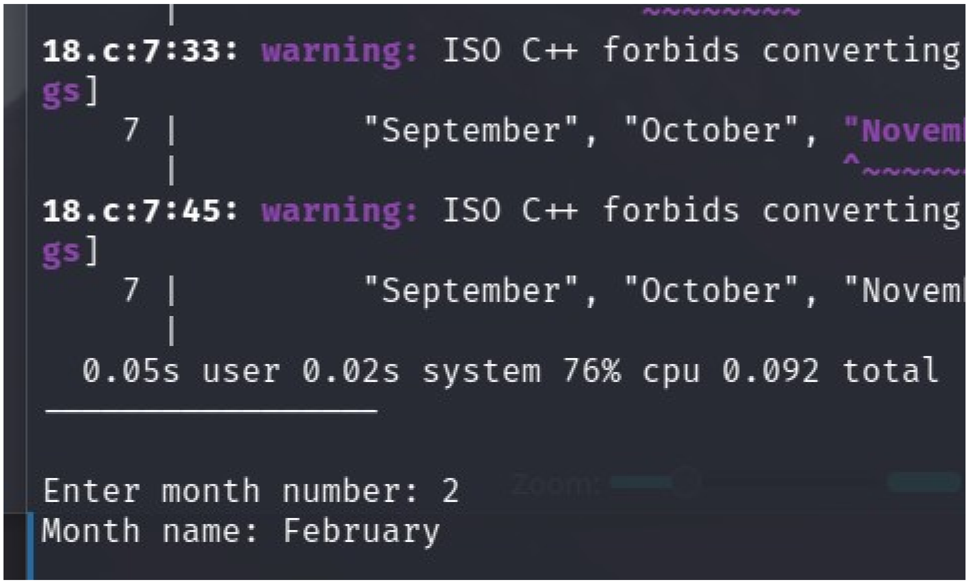
```
#include <stdio.h>

char* locate(char *s1, char *s2, int m)
{
    int i, j;
    static char temp[100];
    for (i=0; i< m; i++)
        temp[i] = s1[i];
    for (j=0; s2[j] != '\0'; j++)
        temp[i+j] = s2[j];
    for (; s1[i] != '\0'; i++)
        temp[i+j] = s1[i];
    return temp;
}

int main()
{
    char s1[100], s2[100];
    int m;
    printf("Enter your string: ");
    scanf("%[^\n]s", s1);

    printf("Enter your string: ");
    scanf(" %[^\n]s", s2);

    printf("Enter your index: ");
    scanf("%d", &m);

    char *updated_string;
    updated_string = locate(s1, s2, m);
    printf("Your string is: %s", updated_string);

    return 0;
}
```



```
compiler didn't create an executable file!


python-docx/input git:main*
(venv) ❯ cl 17.c
compiling 17
  0.05s user 0.01s system 98% cpu 0.068 total


Enter your string: hello
Enter your string: oka
Enter your index: 2
Your string is: heokallo%
```

18 Write a function that takes an integer parameter m representing the month number of the year and returns the corresponding name of the month. For instance, if m = 3, the month is March. Test your program.

```
#include <stdio.h>

char *month_name(int m)
{
    char *months[] = {
        "January", "February", "March", "April", "May", "June", "July", "August",
        "September", "October", "November", "December"};

    return months[m - 1];
}

int main()
{
    int m;

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

    printf("Month name: %s\n", month_name(m));

    return 0;
}
```

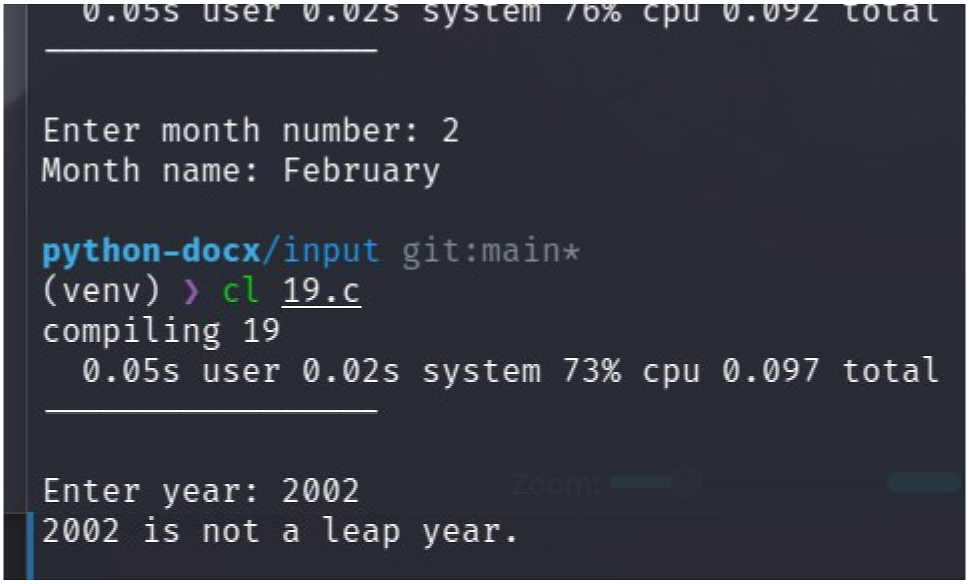


```
      |                                 ~~~~~~~~~
18.c:7:33: warning: ISO C++ forbids converting
gs]
    7 |          "September", "October", "Novem
                                         ^~~~~~~
      |
18.c:7:45: warning: ISO C++ forbids converting
gs]
    7 |          "September", "October", "Novem
      |
  0.05s user 0.02s system 76% cpu 0.092 total
_________________

Enter month number: 2
Month name: February
```

```
}
```

19 In preparing the calendar for a year we need to know whether that particular year is leap year or not. Design a function leap( ) that receives the year as a parameter and returns an appropriate message. What modifications are required if we want to use the function in preparing the actual calendar?

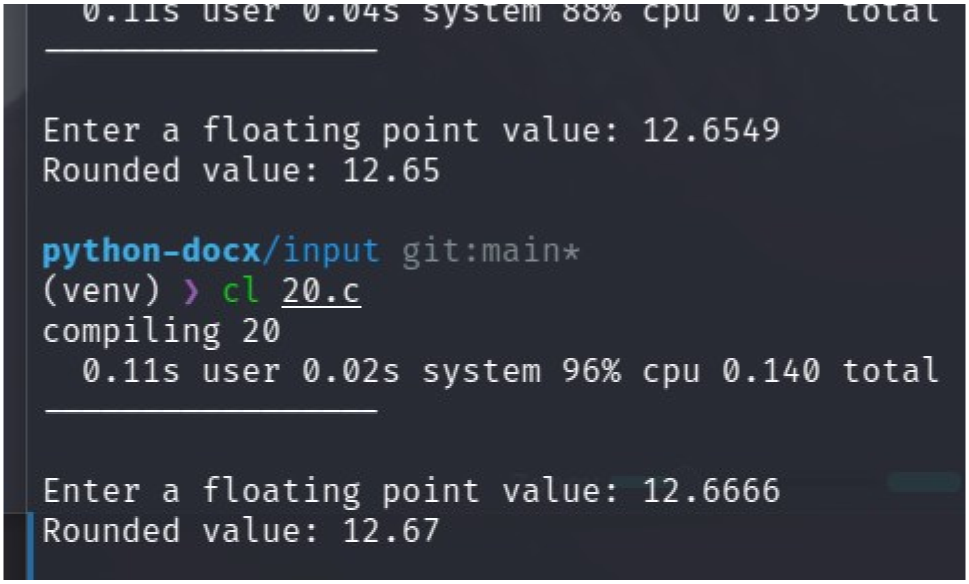
```
#include <stdio.h>

int leap(int year)
{
    if (year % 400 == 0)
        return 1;
    else if (year % 100 == 0)
        return 0;
    else if (year % 4 == 0)
        return 1;
    else
        return 0;
}

int main()
{
    int year;

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

    if (leap(year))
        printf("%d is a leap year.\n", year);
    else
        printf("%d is not a leap year.\n", year);

    return 0;
}
```



```
  0.05s user 0.02s system 76% cpu 0.092 total
─────────────────


Enter month number: 2
Month name: February

python-docx/input git:main*
(venv) ❯ cl 19.c
compiling 19
  0.05s user 0.02s system 73% cpu 0.097 total
─────────────────


Enter year: 2002
2002 is not a leap year.
```

20 Write a function that receives a floating point value x and returns it as a value rounded to two nearest decimal places. For example, the value 123.4567 will be rounded to 123.46 (Hint: Seek help of one of the math functions available in math library).

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

double round2(double x)
{
    return round(x * 100) / 100;
}

int main()
{
    double x;

    printf("Enter a floating point value: ");
    scanf("%lf", &x);

    printf("Rounded value: %.2lf\n", round2(x));

    return 0;
}
```



```
  0.11s user 0.04s system 88% cpu 0.169 total
─────────────────

Enter a floating point value: 12.6549
Rounded value: 12.65

python-docx/input git:main*
(venv) ❯ cl 20.c
compiling 20
  0.11s user 0.02s system 96% cpu 0.140 total
─────────────────

Enter a floating point value: 12.6666
Rounded value: 12.67
```