ព្រះរាជាណាចក្រកម្ពុជា

ជាតិ សាសនា ព្រះមហាក្សត្រ

វិទ្យាស្ថានបច្ចេកវិទ្យាកម្ពុជា

ដេប៉ាតឺម៉ង់ថ្នាក់ឆ្នាំសិក្សាមូលដ្ឋាន

TP10-Function-Part1

Subject: Data strucure & Programming

Student name	ID	score
1.SOUNG VICHKA	e20240672	…..

Teacher: Mr.ROEUN Pachroth

Problem1: Create a function in C programming to display all sequence numbers from 1 to n except the number 10, where n is a parameter of this function.

Example:

INPUT:

Enter a number: 20

Output:

DIsplay number using function displaySequence(1, 20):
1 2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18 19 20

.....

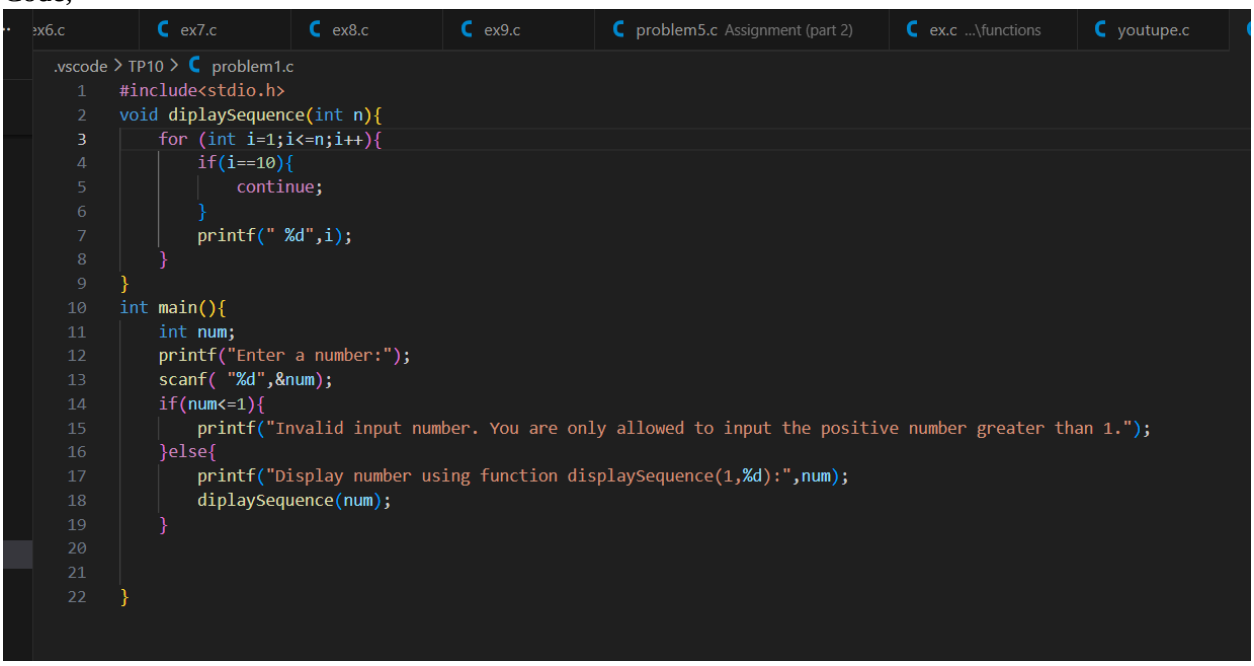
INPUT:

Enter a number: -7

Output:

Invalid input number. You are only allowed to input the positive number greater than 1.

Code;



- **Input**
  - The program asks the user: *"Enter a number:"*
  - The user types a number (stored in num).
- **Validation**
  - If num <= 1, the program prints an error message: *"Invalid input number. You are only allowed to input the positive number greater than 1."*
  - This prevents invalid inputs.
- **Function Call**
  - If the input is valid (num > 1), the program prints a header message: *"Display number using function displaySequence(1,num):"*
  - Then it calls diplaySequence(num).
- **Sequence Printing**
  - Inside diplaySequence(n), a for loop runs from i = 1 to i = n.
  - Each number is printed **except when i == 10** (because of continue).

Result:

```
PS D:\SDP week5>  & 'c:\Users\User\.vscode\extensions\ms-vscode.cpptools-1.29.3-win32-x64\debugAdapters\bin\WindowsDebugLaunch
=Microsoft-MIEngine-In-d1s0xumf.lnk' '--stdout=Microsoft-MIEngine-Out-hwudct20.3yp' '--stderr=Microsoft-MIEngine-Error-nifmxxj
crosoft-MIEngine-Pid-pjwdcfui.m5m' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter a number:10
Display number using function displaySequence(1,10): 1 2 3 4 5 6 7 8 9
PS D:\SDP week5> ^C
PS D:\SDP week5>
PS D:\SDP week5>  & 'c:\Users\User\.vscode\extensions\ms-vscode.cpptools-1.29.3-win32-x64\debugAdapters\bin\WindowsDebugLaunch
=Microsoft-MIEngine-In-ndbvutu4.25j' '--stdout=Microsoft-MIEngine-Out-ogrjwvoy.x4h' '--stderr=Microsoft-MIEngine-Error-34oszxo
crosoft-MIEngine-Pid-htp25t1w.mtv' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter a number:-7
Invalid input number. You are only allowed to input the positive number greater than 1.
PS D:\SDP week5>
```

Problem2: Create a function in C programming to compute a cube of a number n, where n is taken as a parameter of this function. This function has a return value which is the cube of the number n passed though the function parameter. The program asks a user to input the number n, then call the function to compute the cube. Keep the program running as an infinite loop such that the user can test with as many numbers as possible.
Remark: Validate the input number so that it only allows the positive number

Example:

INPUT:

Enter a number: 5

Output:

The cube of 5 equals to: 125

.....

INPUT:

Enter a number: 0

Output:

Error! Invalid input number! You must input a number greater than 0. Please try again!

.....

Code:

part 2) | ex.c ...\functions | youtupe.c | problem1.c ...\TP10 | problem4.c ...\TP10 | ex4.c ...\TP10 | ex.c ...\TP10

.vscode > TP10 > problem2.c

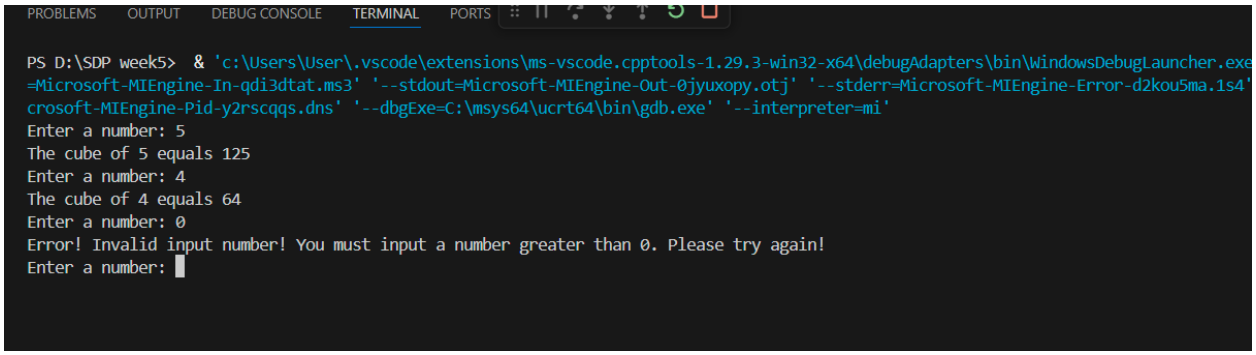
```c
#include <stdio.h>

int computeCube(int n) {
    int cube=n*n*n;
    return cube;
}
int main() {
    int num;
    while (1) {
        printf("Enter a number: ");
        scanf("%d", &num);

        if (num <= 0) {
            printf("Error! Invalid input number! You must input a number greater than 0. Please try again!\n");
        } else {
            printf("The cube of %d equals %d\n", num, computeCube(num));
        }
    }
    return 0;
}
```

- The program repeatedly asks the user to enter a number (because of the while(1) infinite loop).
- If the user enters a number **less than or equal to 0**, it shows an error message telling them to try again.
- If the user enters a valid number (**greater than 0**), it calls the function computeCube(num) which calculates n^3, and then prints the cube of that number.

Result:



.....

Problem3: Design a program to be able to find the diameter, circumference and surface of a circle using a function. The function takes one parameter, i.e., the radius. The function should return three data such as the diameter, the circumference, and the area of this circle.
Diameter = 2 x radius, Circumference = 2 x 3.14159 x radius, Area = 3.14159 x radius x radius;
Hint: Make the type of this function to be a structure consisting of diameter. circumference and area variables.

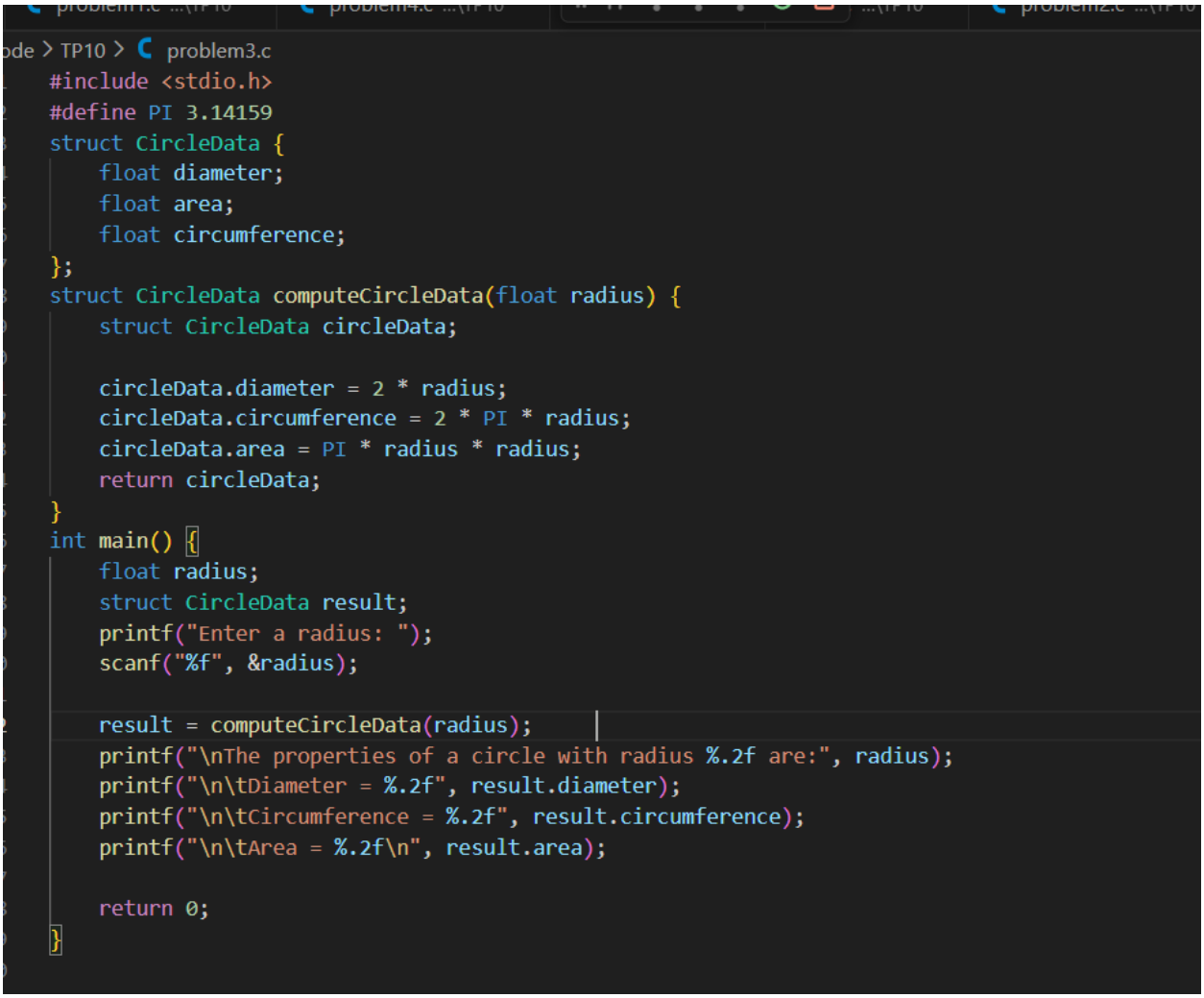
```
struct CircleData{

    float area;
    float circumference;
}
```

The signature (or prototype) of this function is defined as follows:

CircleData **computeDiameterSurface**(float radius){ ..... }

Code:



```
ode > TP10 > problem3.c
#include <stdio.h>
#define PI 3.14159
struct CircleData {
    float diameter;
    float area;
    float circumference;
};
struct CircleData computeCircleData(float radius) {
    struct CircleData circleData;

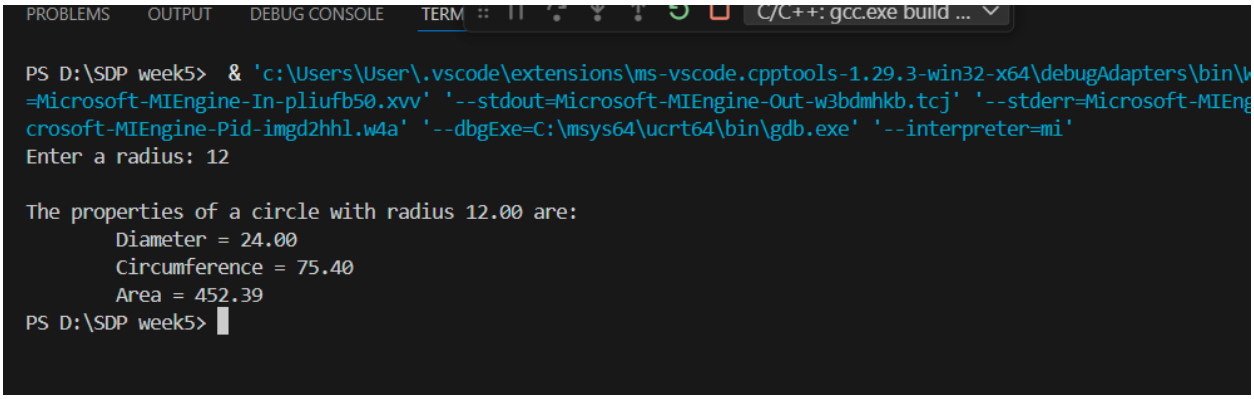
    circleData.diameter = 2 * radius;
    circleData.circumference = 2 * PI * radius;
    circleData.area = PI * radius * radius;
    return circleData;
}
int main() {
    float radius;
    struct CircleData result;
    printf("Enter a radius: ");
    scanf("%f", &radius);

    result = computeCircleData(radius);
    printf("\nThe properties of a circle with radius %.2f are:", radius);
    printf("\n\tDiameter = %.2f", result.diameter);
    printf("\n\tCircumference = %.2f", result.circumference);
    printf("\n\tArea = %.2f\n", result.area);

    return 0;
}
```

- The program asks the user to enter a circle's **radius**.
- It then calls the function computeCircleData(radius) which calculates:
    - **Diameter** = 2\times \mathrm{radius}
    - **Circumference** = 2\times \pi \times \mathrm{radius}
    - **Area** = \pi \times \mathrm{radius^{\mathnormal{2}}}
- These values are stored in a struct CircleData and returned.
- Finally, the program prints out the circle's **diameter, circumference, and area** neatly formatted.

Result:



NAME: SOUNG VICHKA

ID: e20240672

Problem4: Write a C program to ask a user for 10 numbers. Create a function to find the minimum and maximum. Finally display those values to the user on screen.

Example:

INPUT:

Enter 10 numbers: 10 20 -12 9 5 28 11 65 10 -5

Output:

Among the input numbers (10 20 -12 9 5 28 11 65 10 -5), we found that:
The minimum number is: -12, and the maximum number is 65.

Code:

.vscode > TP10 > ex.c

```c
#include <stdio.h>

void findMinMax(int arr[], int size) {
    int min = arr[0];
    int max = arr[0];

    for (int i = 1; i < size; i++) {
        if (arr[i] < min) min = arr[i];
        if (arr[i] > max) max = arr[i];
    }

    printf("The minimum number is:%d", min);
    printf("The maximum number is:%d", max);
}

int main() {
    int arr[10];
    printf("Enter 10 numbers:");

    for (int i = 0; i < 10; i++) {
        scanf("%d", &arr[i]);
    }

    findMinMax(arr, 10);

    return 0;
}
```

```c
void findMinMax(int arr[], int size) {
   int min = arr[0];
   int max = arr[0];

   for (int i = 1; i < size; i++) {
      if (arr[i] < min) min = arr[i];
      if (arr[i] > max) max = arr[i];
   }
```

The function takes an array and its size as input.

- It starts by assuming the first element is both the minimum and maximum.
- Then it loops through the rest of the array:
    - If a number is smaller than min, update min.
    - If a number is larger than max, update max.

int arr[10]; printf("Enter 10 numbers:"); for (int i = 0; i < 10; i++) { scanf("%d", &arr[i]); }

- Declares an array of size 10.
- Prompts the user to enter 10 numbers.
- Uses a loop to read each number into the array.

findMinMax(arr, 10);

- Passes the array and its size to findMinMax.
- The function then processes the array and prints the min and max.

```
PS D:\SDP week5>  & 'c:\Users\User\.vscode\extensions\ms-vscode.cpptools-1.29.3-win32-x64\debugAdapters\bin\WindowsDebugL
--stdin=Microsoft-MIEngine-In-sus35prr.rju' '--stdout=Microsoft-MIEngine-Out-12c1ppko.n5w' '--stderr=Microsoft-MIEngine-E
r4f' '--pid=Microsoft-MIEngine-Pid-ulxs5o15.2dk' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter 10 numbers:1 2 3 4 5 6 7 8 9 10
The minimum number is: 1The maximum number is: 10
PS D:\SDP week5>
```

Problem5: Create a C program to solve the quadratic equation ax^2+bx+c=0 using a function. The function takes 3 parameters which are the coefficients of the equation such as a, b and c. This function returns some structure data such as status code (1: the equation has double root, 2: the equation has 2 roots, 3: the equation has complex roots), root x1, and root x2.

```
struct Root{
        int status_code;
        float x1;
        float x2;
}
```

The signature of this function is defined by: Root **solveQuadratic**(int a, int b, int c)

Deta = b*b – 4ac

1. If deta > 0
   - Status = 1
   - X1 = (-b + sqrt(deta)) / 2a
   - X2 = (-b - sqrt(deta)) / 2a
2. If deta = 0
   - Status = 2
   - X1 = X2 = -b/2a
3. If deta < 0
   - Status = 3
   - X1 = -b/2a
   - X2 = sqrt(-deta) / 2a

.vscode > TP10 > probem5.c

```c
#include <stdio.h>
#include <math.h>
struct Root {
    int status_code;
    float x1,x2;
};
struct Root solveQuadratic(int a, int b, int c) {
    struct Root root;
    int delta = b * b - 4 * a * c;
    if (delta > 0) {
        root.status_code = 1;
        root.x1 = (-b + sqrt(delta)) / (2.0 * a);
        root.x2 = (-b - sqrt(delta)) / (2.0 * a);
    }
    else if (delta == 0) {
        root.status_code = 2;
        root.x1 = root.x2 = -b / (2.0 * a);
    }
    else {
        root.status_code = 3;
        root.x1 = -b / (2.0 * a);
        root.x2 = sqrt(-delta) / (2.0 * a);
    }
    return root;
}
int main() {
    struct Root result;
    int a, b, c;
    printf("Enter a, b, c: ");
    scanf("%d %d %d", &a, &b, &c);
    result = solveQuadratic(a, b, c);
    if (result.status_code == 1) {
        printf("X1 = %.2f\n", result.x1);
        printf("X2 = %.2f\n", result.x2);
    }
    else if (result.status_code == 2) {
        printf("X = %.2f\n", result.x1);
    }
    else {
        printf("X1 = %.2f + %.2fi\n", result.x1, result.x2);
        printf("X2 = %.2f - %.2fi\n", result.x1, result.x2);
    }
    return 0;
}
```

- **Struct Definition**

```
 struct Root {
int status_code;
float x1, x2;
  };
```

  - Holds the results:
    - status_code → tells which case we're in (two real roots, one real root, or complex roots).
    - x1, x2 → the actual roots.

Discriminant Calculation

- int delta = b * b - 4 * a * c;

- This is the discriminant (\Delta).
- It decides the type of roots.

➢ **Cases**

- **Case 1: Δ > 0** → two distinct real roots
- root.x1 = (-b + sqrt(delta)) / (2.0 * a);
- root.x2 = (-b - sqrt(delta)) / (2.0 * a);
- root.status_code = 1;
- **Case 2: Δ = 0** → one real root (double root)
- root.x1 = root.x2 = -b / (2.0 * a);
- root.status_code = 2;
- **Case 3: Δ < 0** → complex roots
- root.x1 = -b / (2.0 * a); // real part
- root.x2 = sqrt(-delta) / (2.0 * a); // imaginary part
- root.status_code = 3;

➢ **Main Function**

- Reads a, b, c from the user.
- Calls solveQuadratic(a, b, c).
- Prints the roots depending on the status_code.

Result:

