Experiment 1

1. Calculation of the area and perimeter of geometric shapes (70 pt)

Write a C code that calculates the area and perimeter of geometric shapes such as a circle, square, and equilateral triangle.

Your code must also include the following items:

- The characters 'c' for a circle, 's' for a square, and 't' for an equilateral triangle should be chosen by users.

- The radius for the circle, and the side length for the triangle and square, should be entered by the user.

- The area and perimeter of the selected shape should be displayed with three leading zeros for the integer part and four digits for the decimal part

Some output examples are shown below: $(\pi = 3.14159)$

for a circle	Please select a shape (c for circle, s for square, t for equilateral triangle): c Enter the radius of the circle: 3 Circle Area: 00028.2743 Circle Circumference: 00018.8495
for a square	Please select a shape (c for circle, s for square, t for equilateral triangle): s Enter the side length of the square: 3 Square Area: 0009.0000 Square Perimeter: 00012.0000
for a triangle	Please select a shape (c for circle, s for square, t for equilateral triangle): t Enter the side length of the equilateral triangle: 3 Equilateral Triangle Area: 0003.8971 Equilateral Triangle Perimeter: 0009.0000
else	Please select a shape (c for circle, s for square, t for equilateral triangle): a You made an invalid choice.

2. ASCII code operations (30 pt)

Write a C code that lists the ASCII codes of single-digit numbers using a loop and calculates the average of these codes.

The output of your program should be as follows:

```
ASCII codes of single-digit numbers:
'0' -> ASCII code: 48
'1' -> ASCII code: 49
'2' -> ASCII code: 50
'3' -> ASCII code: 51
'4' -> ASCII code: 52
'5' -> ASCII code: 53
'6' -> ASCII code: 54
'7' -> ASCII code: 55
'8' -> ASCII code: 56
'9' -> ASCII code: 57
Average of ASCII codes for single-digit numbers: 52.50
```

Hint:

for Loop:

```c
int i;
for (i = initial_value; i <= final_value; i++) {
    /* Code to be repeated */
    // ...
    // ...
}
```

while Loop:

```c
int i = initial_value;
while (i <= final_value) {
    /* Code to be repeated */
    // ...
    // ...

    /* Update 'i' to prevent an infinite loop */
    i = i + 1;
    // OR
    // i++;
}
```

if Statement:

```c
if (condition1) {
    // Code to be executed if condition1 is true
} else if (condition2) {
    // Code to be executed if condition1 is false and condition2 is true
} else {
    // Code to be executed if neither condition1 nor condition2 is true
}
```