If you want to contribute to these practice exercises, you can send your contribution to hafiz.hamza@nu.edu.pk

For more practice questions and reference material:

- C++ Programming: Program Design Including Data Structures, by D. S. Malik
- C++: How to Program, by Deitle & Deitle
- Learn C++
- GeeksforGeeks

1. Simple Problems

Exercise 1.1 (A)

Given the length of the side of a square, calculate and print the area of the square.

Sample:

```
Enter length: 5
Area: 25
```

Exercise 1.1 (B)

Given the length of the side of a square, calculate and print the area of the square in the given format.

Sample:

```
Enter length: 5
The area of square with length 5 is 25
```

Exercise 1.2

Write a C++ program which converts a Celsius temperature into Fahrenheit.

Sample:

```
Enter temperature in Celsius: 44.6
Temperature in Fahrenheit: 112.28
```

Exercise 1.3 (A)

Write a C++ program to swap two variables.

Sample:

```
Enter value for first variable: 7
Enter value for second variable: 19

Value of first variable: 19
Value of second variable: 7
```

Exercise 1.3 (B)

Write a C++ program to swap two variables without using a third variable.

Sample:

```
Enter value for first variable: 7
Enter value for second variable: 19

Value of first variable: 19
Value of second variable: 7
```

2. If/Else

Exercise 2.1 (A)

Given two numbers, print the larger of them. Assume that the two numbers are distinct.

Sample:

```
Enter first number: 17
Enter second number: 65
65 is greater than 17
```

Exercise 2.1 (B)

Given two numbers, print the larger of them. This time, the numbers can be equal.

Sample:

```
Enter first number: 1335
Enter second number: 987
1335 is greater than 987
```

Sample:

```
Enter first number: 144
Enter second number: 144
Both numbers are equal
```

Exercise 2.2

Given three numbers, print the largest of them. Assume that the three numbers are distinct.

Sample:

```
Enter first number: 64
Enter second number: 81
Enter third number: 49
81 is greatest
```

Sample:

```
Enter first number: 400
Enter second number: 361
Enter third number: 289
400 is greatest
```

Sample:

```
Enter first number: 100
Enter second number: 100
Enter third number: 100
All three numbers are equal
```

Solution:

```
int main() {

	int num1, num2, num3;

	cout << "Please enter first number: ";
	cin >> num1;

	cout << "Please enter second number: ";
	cin >> num2;

	cout << "Please enter third number: ";
	cin >> num3;

	if (num1 > num2) {
		if (num1 > num3) {
			cout << num1 << " is greatest\n";
		}
	}

	if (num2 > num1) {
		if (num2 > num3) {
			cout << num2 << " is greatest\n";
		}
	}

	if (num3 > num1) {
		if (num3 > num2) {
			cout << num3 << " is greatest\n";
		}
	}

	return 0;
}
```

Exercise 2.3

Given a number, tell whether it's even or odd.

Sample:

```
Enter number: 9
9 is an odd number
```

Exercise 2.4

Given a number, tell whether it's divisible by 3.

Sample:

```
Enter number: 11
11 is not divisible by 3
```

Sample:

```
Enter number: 36
36 is divisible by 3
```

Exercise 2.5

Given two numbers x and y, tell whether y completely divides x i.e. x/y gives an integer.

Sample:

```
Enter X: 121
Enter Y: 11
121 is divisible by 11
```

Sample:

```
Enter X: 56
Enter Y: 3
56 is not divisible by 3
```

Exercise 2.6

Given three sides of a triangle, tell whether the triangle is right-angled.

Sample:

```
Enter first side: 0.8
Enter second side: 1
Enter third side: 0.6

The triangle is right-angled
```

Sample:

Enter first side: **13**
Enter second side: **8**
Enter third side: **15**

The triangle is not right-angled

3. Loops

Exercise 3.1

Write a C++ program to print all natural numbers from 1 to n - using while loop

Exercise 3.2

Write a C++ program to print all natural numbers in reverse (from n to 1) - using while loop

Exercise 3.3

Write a C++ program to print all alphabets from a to z - using while loop

Exercise 3.4

Write a C++ program to print all even numbers between 1 to 100 - using while loop

Exercise 3.5

Write a C++ program to print all odd numbers between 1 to 100.

Exercise 3.6

Write a C++ program to find the sum of all natural numbers from 1 to n.

Exercise 3.7

Write a C++ program to find the sum of all even numbers from 1 to n.

Exercise 3.8

Write a C++ program to find the sum of all odd numbers from 1 to n.

Exercise 3.9

Write a C++ program to print a multiplication table of any number.

Exercise 3.10

Write a C++ program to count the number of digits in a number.

Exercise 3.11

Write a C++ program to find the first and last digit of a number.

Exercise 3.12

Write a C++ program to find the sum of the first and last digits of a number.

Exercise 3.13

Write a C++ program to swap first and last digits of a number.

Exercise 3.14

Write a C++ program to calculate the sum of digits of a number.

Exercise 3.15

Write a C++ program to calculate the product of digits of a number.

Exercise 3.16

Write a C++ program to print a number in reverse.

Solution:

```
int main()
{
	int input;
	cout << "Enter number: ";
	cin >> input;

	int reverse = 0;
	while (input != 0) {
		reverse = reverse * 10 + input % 10;
		input = input / 10;
	}
	cout << reverse << endl;
	return 0;
}
```

Exercise 3.17

Write a C++ program to check whether a number is palindrome or not.

Exercise 3.18

Write a C++ program to find frequency of each digit in a given integer.

Exercise 3.19

Write a C++ program to enter a number and print it in words.

Exercise 3.20

Write a C++ program to print all ASCII characters with their values.

Exercise 3.21

Write a C++ program to find power of a number using *for* loop.

Exercise 3.22

Write a C++ program to find all factors of a number.

Exercise 3.23

Write a C++ program to calculate the factorial of a number.

Exercise 3.24

Write a C++ program to find HCF (GCD) of two numbers.

Exercise 3.25

Write a C++ program to find LCM of two numbers.

Exercise 3.26 (A)

Write a C++ program to check whether a number is Prime.

Solution:

```
int main()
{
	int N;

	cout << "Enter N: ";
	cin >> N;

	int i = 2;
	for (i = 2; i < N; ++i) {
		if (N % i == 0) {
			cout << N << " isn't a prime number\n";
			break;
		}
	}
	if (i == N)
		cout << N << " is a prime number\n";

	return 0;
}
```

Exercise 3.26 (B)

Write a C++ program to print all prime numbers between M and N.

Solution:

```
int main()
{
	int M, N;
	cout << "Enter M: ";
	cin >> M;
	cout << "Enter N: ";
	cin >> N;

	for (int i = M; i <= N; ++i) {
		int j;
		for (j = 2; j <= i - 1; ++j) {
			if (i % j == 0) {
				break;
			}
		}
		if (j == i)
			cout << i << endl;
	}
```

```
        return 0;
}
```

Exercise 3.27

Print the following pattern.

```
*****
*   *
*   *
*   *
*****
```

Hollow Square Pattern

Exercise 3.28

Print the following pattern.

```
*****
 *****
  *****
   *****
    *****
```

Rhombus Pattern

Exercise 3.29

Print the following pattern.

```
*****
 *   *
  *   *
   *   *
    *****
```

Hollow Rhombus Pattern

Exercise 3.30

Print the following pattern.

```
    *****
   *****
  *****
 *****
*****
```

Mirrored Rhombus Pattern

Exercise 3.31

Print the following pattern.

```
    *****
   *   *
  *   *
 *   *
*****
```

Hollow Mirrored Rhombus Pattern

Exercise 3.32

Print the following pattern.

```
*
**
***
****
*****
```

Right Triangle Pattern

Solution:

```
int main()
{
	int input;
	cout << "Enter height of triangle: ";
	cin >> input;

	for (int i = 0; i < input; ++i) {
		for (int j = 0; j <= i; ++j) {
			cout << "*";
		}
		cout << endl;
	}
	return 0;
}
```

Exercise 3.33

Print the following pattern.

```
*
**
* *
*  *
*****
```

Hollow Right Triangle Pattern

Exercise 3.34

Print the following pattern.

```
    *
   **
  ***
 ****
*****
```

Mirrored Right Triangle Pattern

Solution:

```
int main()
{
	int input;
	cout << "Enter height of triangle: ";
	cin >> input;

	for (int i = 0; i < input; ++i) {
		for (int j = 0; j < input - i - 1; ++j) {
			cout << " ";
		}
		for (int j = 0; j <= i; ++j) {
			cout << "*";
		}
		cout << endl;
	}
	return 0;
}
```

Exercise 3.35

Print the following pattern.

```
    *
   **
  * *
 *  *
*****
```

Hollow Mirrored Right Triangle Pattern

Exercise 3.36

Print the following pattern.

```
*****
****
***
**
*
```

Inverted Right Triangle Pattern

Exercise 3.37

Print the following pattern.

```
*****
*   *
* *
**
*
```

Hollow Inverted Right Triangle Pattern

Exercise 3.38

Print the following pattern.

```
*****
 ****
  ***
   **
    *
```

Inverted Mirrored Right Triangle Pattern

Exercise 3.39

Print the following pattern.

```
*****
 *   *
  * *
   **
    *
```

Hollow Inverted Mirrored Right Triangle Pattern

Exercise 3.41

Print the following pattern.

```
    *
   ***
  *****
 *******
```

```
*********
```

Pyramid Star Pattern

Solution:

```
int main()
{
	int input;
	cout << "Enter height of pyramid: ";
	cin >> input;

	for (int i = 0; i < input; ++i) {
		for (int j = 0; j < input - i - 1; ++j) {
			cout << " ";
		}
		for (int j = 0; j < (2 * i + 1); ++j) {
			cout << "*";
		}
		cout << endl;
	}
	return 0;
}
```

Exercise 3.42

Print the following pattern.

```
    *
   * *
  *   *
 *     *
*********
```

Hollow Pyramid Pattern

Exercise 3.43

Print the following pattern.

```
*********
 *******
```

```
  *****
   ***
    *
```

Inverted Pyramid Pattern

Exercise 3.44

Print the following pattern.

```
*********
 *      *
  *    *
   * *
    *
```

Hollow Inverted Pyramid Pattern

Exercise 3.45 (A)

Print the following pattern.

```
*
**
***
****
*****
****
***
**
*
```

Half Diamond Pattern

Solution:

```
int main()
{
	int input;
	cout << "Enter radius of diamond: ";
	cin >> input;

	for (int i = 1; i <= input; ++i) {
		for (int j = 1; j <= i; ++j) {
			cout << "*";
		}
		cout << endl;
	}

	for (int i = input - 1; i >= 1; --i) {
		for (int j = 1; j <= i; ++j) {
			cout << "*";
		}
		cout << endl;
	}
	return 0;
}
```

Exercise 3.45 (B)

Print the following pattern. The max number of ***for*** loops that you can use is **2**.

```
*
**
***
****
*****
****
***
**
*
```

`Half Diamond Pattern`

Solution:

```
int main()
{
	int input;
	cout << "Enter radius of diamond: ";
	cin >> input;

	for (int i = 1, counter = 0; i >= 1; counter++) {
		for (int j = 1; j <= i; ++j) {
			cout << "*";
		}
		if (counter < input)
			++i;
		else
			--i;
		cout << endl;
	}

	return 0;
}
```

Exercise 3.46

Print the following pattern.

```
    *
   **
  ***
 ****
*****
 ****
  ***
   **
    *
```

Mirrored Half Diamond Pattern

Exercise 3.47 (A)

Print the following pattern.

```
    *
   ***
  *****
 *******
```

```
*********
 *******
  *****
   ***
    *
```

Diamond Pattern

Exercise 3.47 (B)

Print the following pattern. The max number of ***for*** loops that you can use is **3**.

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

Diamond Pattern

Solution:

```
int main()
{
	int input;
	cout << "Enter diameter of the diamond: ";
	cin >> input;

	for (int i = 1, counter = 0; i >= 1; ++counter) {
		for (int j = 1; j <= input / 2 - i + 1; ++j)
			cout << " ";

		for (int j = 1; j <= (2 * i - 1); ++j)
			cout << "*";

		cout << endl;

		if (counter < input / 2)
			++i;
		else
			--i;
```

```
        }
        return 0;
}
```

Exercise 3.48

Print the following pattern.

```
**********
****  ****
***    ***
**      **
*        *
*        *
**      **
***    ***
****  ****
**********
```

Hollow Diamond Pattern

Exercise 3.49

Print the following pattern.

```
*****
  ****
    ***
      **
        *
      **
    ***
  ****
*****
```

Right Arrow Pattern

Exercise 3.50

Print the following pattern.

```
    *****
   ****
  ***
 **
*
 **
  ***
   ****
    *****
```

Left Arrow Pattern

Exercise 3.51

Print the following pattern.

```
    +
    +
    +
    +
+++++++++
    +
    +
    +
    +
```

Plus Pattern

Exercise 3.52

Print the following pattern.

```
*       *
 *     *
  *   *
   * *
    *
   * *
  *   *
 *     *
*       *
```

X Pattern

Exercise 3.53

Print the following pattern.

```
 ***
*   *
*   *
*   *
 ***
*   *
*   *
*   *
 ***
```

Eight Pattern

Exercise 3.54

Print the following pattern.

```
12345
23451
34521
45321
54321
```

Exercise 3.55

Print M x N multiplication table. For example,

```
1  2  3  4  5  6  7  8  9 10
2  4  6  8 10 12 14 16 18 20
3  6  9 12 15 18 21 24 27 30
4  8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
```

Solution:

```
int main()
{
	int M, N;
	cout << "Enter M: ";
	cin >> M;
	cout << "Enter N: ";
	cin >> N;

	for (int i = 1; i <= N; ++i) {
		for (int j = 1; j <= M; ++j) {
			cout << setw(3) << i * j;
		}
		cout << endl;
	}
	return 0;
}
```

Exercise 3.56

Print the following pattern.

```
5 5 5 5 5 5 5 5 5
5 4 4 4 4 4 4 4 5
5 4 3 3 3 3 3 4 5
5 4 3 2 2 2 3 4 5
5 4 3 2 1 2 3 4 5
5 4 3 2 2 2 3 4 5
5 4 3 3 3 3 3 4 5
5 4 4 4 4 4 4 4 5
5 5 5 5 5 5 5 5 5
```

Exercise 3.57

Print the following pattern.

```
1    2    3    4    5
16   17   18   19   6
15   24   25   20   7
14   23   22   21   8
13   12   11   10   9
```

Exercise 3.58

Print the following pattern.

```
1         1
12       21
123     321
1234   4321
1234554321
```

Exercise 3.59

Goldbach's conjecture is one of the oldest and best-known unsolved problems in number theory and all of mathematics. It states: Every even integer greater than 2 is the sum of two primes.

Write a program which takes a number N from the user. The program then takes N even integers greater than 2 from the user one by one and displays every integer as a sum of two primes.

4. Dry Run

Exercise 4.1

Dry run the following lines of code. Each statement is performed independently and not connected with the previous lines.

a = 2, b = 20, c =10, d =5

1. e = (a + b) * c / d;

cout << ”Value of (a + b) * c / d is:” << e << endl;

2. e = ((a + b) * c) / d;

cout << ”Value of ((a + b) * c) / d is:” << e << endl;

3. e = (a + b) * (c / d);

cout << ”Value of (a + b) * (c / d) is:” << e << endl;

4. e = a + (b * c) / d;

cout << ”Value of a + (b * c) / d is:” << e << endl;

5. e = a + (b - c) / d * c + a;

cout << ”Value of a + (b - c) / d * c + a is:” << e << endl;

6. cout << ”Value of c++ and ++c is:” << c++ << ++c << endl;

7. cout << ”Value of ++c + a++ is:” << ++c + a++ << endl;

8. cout << ”Value of d++ % c++ is:” << d++ % c++ << endl;

9. cout << ”Value of c++ << c++ << c++ << c++ is:” << c++ << c++ << c++ << c++ << endl;

10. if ((c = (d == 9 | | c == 4)))

    cout << “Value of c is:” << c << endl;

Exercise 4.2

What will the following program print on screen?

```
#include<iostream>
using namespace std;

int main()
{
	int x = 5;
	int y = 6;
	cout << “Result = “ << ++x * y << endl;
	cout << “x = “ << x << endl;
	return 0;
}
```

Exercise 4.3

What will the following program print on screen?

```
#include<iostream>
using namespace std;

int main()
{
	int x = 5;
	int y = 6;
	cout << “Result = “ << x++ * y << endl;
	cout << “x = “ << x << endl;
	return 0;
}
```

Exercise 4.4

What will the following program print on screen?

```
#include<iostream>
using namespace std;

int main()
{
	int x = 5;
	int y = 6;
	cout << “Result = “ << ++(x+y) << endl;
	cout << “x = “ << x << endl;
	return 0;
}
```

Exercise 4.5

What will the following program print on screen?

```
#include<iostream>
using namespace std;

int main()
{
	int x = 30 / 5 * 4 + 3 * 6 / 2;
	cout << “x = “ << x << endl;
	return 0;
}
```

Exercise 4.6

What will the following program print on screen?

```
#include<iostream>
using namespace std;

int main()
{
```

```
    int x = 3 * 6 / 2 + 30 / 6 * 2;
    cout << "x = " << x << endl;
    return 0;
}
```

Exercise 4.7

What will the following program print on screen?

```
#include<iostream>
using namespace std;

int main()
{
    int x = 5;
    cout << "Result 1 = " << (x == 5) << endl;
    cout << "Result 2 = " << (x > 10) << endl;
    cout << "Result 3 = " << (x < 10) << endl;
    cout << "Result 4 = " << (x <= 5) << endl;

    return 0;
}
```

Exercise 4.8

What will the following program print on screen?

```
#include<iostream>
using namespace std;

int main()
{
    int x = 5;
    bool isValid = true;

    cout << "Result 1 = " << ((x == 5) && !isValid) << endl;
    cout << "Result 2 = " << ((x > 10) && isValid) << endl;
    cout << "Result 3 = " << (((x == 5) && !is Valid) || (x < 10) && isValid)  << endl;
    cout << "Result 4 = " << (((x != 5) || !is Valid) && (x == 5) && isValid)  << endl;

    return 0;
}
```

Exercise 4.9 (Pass-by-value and Pass-by-reference)

What will be the output of the following program?

```
#include<iostream>
using namespace std;

int GuessWhat(int& a, int& b) {
	int c = 0;
	a = c+1;
	b = c+2;
	return a + b;
}

int GuessAgain(int& a, int &b, int c, int d) {

	int i;
	c = c + 5;
	d = d * 2;
	for (i = 0; i < c; ++i)
		a = a+d;
	b = c + d;
	return i;
}

int main() {
	int a = 0, b = 0, c = 0, d = 0,e = 0;

	e = GuessWhat(a,b);
	cout << "The 1st value of E is " << e;

	e = GuessWhat(d,c);
	cout << "\nThe 2nd value of E is " << e;

	e = GuessAgain(a,b,c,d);
	cout << "\nThe 3rd value of E is " << e;

	e = GuessAgain(d,c,b,a);
	cout << "\nThe 4th value of E is " << e;

	cout << endl << endl;
	return 0;
}
```

5. Functions

Exercise 5.1

Write a C++ function to print a right-angled triangle of a specific height.

```
*
**
***
****
*****

Right Triangle Pattern
```

Solution:

```
void printTriangle(int height) {

	for (int i = 0; i < height; ++i) {
		for (int j = 0; j <= i; ++j) {
			cout << "*";
		}
		cout << endl;
	}
}
```

Exercise 5.2

Write a C++ function to check whether a number is prime or not.

Solution:

```
bool isPrime(int N) {

	for (int i = 2; i < N; ++i) {
		if (N % i == 0)
			return false;
	}
	return true;
}
```

Exercise 5.3

Write a C++ function to calculate and return **x power y** using a *for* loop.

Exercise 5.4

Write a C++ function to swap two variables.

Solution:

```
void swap(int& x, int& y) {
	int temp = x;
	x = y;
	y = temp;
}
```

Exercise 5.5

Write a C++ function which receives two numbers and an operator (a character from any of the following symbols: '+', '-', '*', '/' and '%') and returns the answer of the operator applied on the given two numbers.

Exercise 5.6

Given five points (where each point has an x and y coordinate) where four points represent the four corners of a rectangular cage and the 5th point is just the location of a prisoner. Write a C++ function which receives all these points as parameters and return true if the prisoner is inside the cage or false if the prisoner is outside the cage.

Exercise 5.7

Write a C++ function which receives four points (where each point has an x and y coordinate) as parameters and tells whether these points are the coordinates of a square, rectangle or just a quadrilateral.

Exercise 5.8

Write a C++ program to simulate a simple **Rock Paper Scissors** game. In this game, two players simultaneously say (or display a hand symbol representing) either rock, paper or scissors. The winner is the one whose choice dominates the other. The rules are: paper dominates (wraps) rock, rock dominates (breaks) scissors and scissors dominate (cut) paper. You can use r/R for rock, p/P for paper and s/S scissors.

1. Design a function which takes as argument 2 character symbols (choices of Player 1 and Player 2) and returns the result 0,1 or 2 (0 means game draw, 1 means Player 1 wins and 2 means Player 2 wins).
2. In *main()*, simulate this game as many times as the user wants by calling the above function in a *while* loop. Whenever the user wants to finish playing, he'll enter -1.
3. In the end, your program should tell how many times the game was played and win count for each player.

6. Arrays

Exercise 6.1

Write a C++ program to take 5 integers from the user and print them in reverse order.

Solution:

```
int main() {
	const int SIZE = 5;
	int myArray[SIZE];

	for (int i = 0; i <SIZE ;++i) {
		cout << "Enter value# " << i + 1 << ": ";
		cin >> myArray[i];
	}
	for (int i = SIZE-1; i >=0; --i)
		cout << myArray[i]<<"  ";
	return 0;
}
```

Exercise 6.2 (Linear Search)

Write a C++ program to take 10 integers and another integer as a key from the user. Your program must tell whether the key exists in those 10 integers.

Solution:

```
int main() {
	const int SIZE = 10;
	int myArray[SIZE];

	for (int i = 0; i <SIZE ;++i) {
		cout << "Enter value# " << i + 1 << ": ";
		cin >> myArray[i];
	}

	int key;
	cout << "Enter value to search: ";
	cin >> key;

	int i;
	for (i = 0; i < SIZE; ++i)
		if (key == myArray[i])
			break;
	if (i < SIZE)
		cout << key << " found at index# " << i << "\n";
	else
		cout << key << " not found\n";

	return 0;
}
```

Exercise 6.3 (Array Reversal)

Write a C++ program to input 10 integers in an array and then reverse the array.

Solution:

```
int main() {
	const int SIZE = 10;
	int myArray[SIZE];

	// Taking input in array
	for (int i = 0; i <SIZE ;++i) {
		cout << "Enter value# " << i + 1 << ": ";
		cin >> myArray[i];
	}

	// Array reversal
	for (int i = 0, j=SIZE-1; i < j; ++i, --j)
		swap(myArray[i], myArray[j]);

	// Printing array
	for (int i = 0; i < SIZE; ++i)
		cout << myArray[i]<<"  ";

	return 0;
}
```

Exercise 6.4

Write a C++ program to take 10 integers in an array and another integer as a key from the user. Your program must tell how many times the key occurs in the array.

Exercise 6.5 Take input in matrices from a file to ease the process.

Write a C++ program to take 10 integers in an array from the user. Your program must display the frequency of every integer in the array.

7. C-Strings (Character Arrays)

Exercise 7.1 (String Length)

Custom code

```
int i=0;
for (; string1[i] != '\0'; ++i);

cout << "Length: " << i << endl;
```

C++ provided function

```
cout << "Length: " << strlen(string1); << endl;
```

Exercise 7.2 (String Copy)

Custom code

```
for (int i = 0; i <= strlen(string1); ++i)
        string1[i] = string2[i];
```

C++ provided function

```
strcpy_s(string1,string2);
```

Exercise 7.3 (String Compare)

Custom code

C++ provided function

```
int result = strcmp(string1, string2);
if (result == 0)
        cout << "The strings are same\n";
```

```
else if (result == 1)
	cout << "string1 is greater than string2\n";
else
	cout << "string2 is greater than string1\n";
```

Exercise 7.4 (String Concatenation (append))

Custom code

```
for (int i = strlen(string1), j = 0; j <= strlen(string2); ++j, ++i)
	string1[i] = string2[j];
```

C++ provided function

```
strcat_s(string1,string2);
```

Exercise 7.5 (Substring Finding)

```
int findSubstr(const char str[], const char keyword[]) {
	for (int i = 0; i < int(strlen(str) - strlen(keyword)) - 1; ++i) {
		int j;
		for (j = 0; j < strlen(keyword); ++j)
			if (keyword[j] != str[i + j])
				break;

		if (j == strlen(keyword))
			return i;
	}
	return -1;
}
```

8. 2D Arrays

Exercise 8.1 (A)

Write a C++ program to initialize an integer array and output it **row wise**. In addition, display the sum of each row.

Solution:

```
int main() {
	const int ROWS = 5;
	const int COLS = 3;

	int stock[ROWS][COLS] = { {1,2,3},
				  {4,5,6} ,
				  {7,8,9} ,
				  {2,3,4} ,
				  {5,6,7} };

	for (int i = 0; i < ROWS; ++i) {
		int sum = 0;
		for (int j = 0; j < COLS; ++j) {
			sum += stock[i][j];
			cout << stock[i][j] << " ";
		}
		cout << "Sum: " << sum;
		cout << endl;
	}

	return 0;
}
```

Exercise 8.1 (B)

Write a C++ program to initialize an integer array and output it **column wise**. In addition, display the sum of each column.

Solution:

```
int main() {
	const int ROWS = 5;
	const int COLS = 3;

	int stock[ROWS][COLS] = { {1,2,3},
				  {4,5,6} ,
				  {7,8,9} ,
				  {2,3,4} ,
				  {5,6,7} };

	for (int i = 0; i < COLS; ++i) {
		int sum = 0;
		for (int j = 0; j < ROWS; ++j) {
			sum += stock[j][i];
			cout << stock[j][i] << " ";
		}
		cout << "Sum: " << sum;
		cout << endl;
	}

	return 0;
}
```

Exercise 8.2

Write a C++ program for matrix addition. Take input in matrices from a file to ease the process.

Solution:

```
int main() {
	const int ROWS = 2, COLS = 3;

	int matrix1[ROWS][COLS];
	int matrix2[ROWS][COLS];
	int result[ROWS][COLS];

	ifstream fin("mat_add1.txt");
	for (int i = 0; i < ROWS; ++i)
		for (int j = 0; j < COLS; ++j)
			fin >> matrix1[i][j];
```

```
        fin.close();

        cout << "\n  Matrix 1\n";
        for (int i = 0; i < ROWS; ++i) {
                for (int j = 0; j < COLS; ++j)
                        cout << matrix1[i][j] << " ";
                cout << endl;
        }

        fin.open("mat_add2.txt");
        for (int i = 0; i < ROWS; ++i)
                for (int j = 0; j < COLS; ++j)
                        fin >> matrix2[i][j];
        fin.close();

        cout << "\n  Matrix 2\n";
        for (int i = 0; i < ROWS; ++i) {
                for (int j = 0; j < COLS; ++j)
                        cout << matrix2[i][j] << " ";
                cout << endl;
        }

        for (int i = 0; i < ROWS; ++i)
                for (int j = 0; j < COLS; ++j)
                        result[i][j] = matrix1[i][j] + matrix2[i][j];

        cout << "\n  Result\n";
        for (int i = 0; i < ROWS; ++i) {
                for (int j = 0; j < COLS; ++j)
                        cout << result[i][j] << " ";
                cout << endl;
        }
        return 0;
}
```

Exercise 8.3 (Matrix Multiplication)

Write a C++ program for matrix multiplication. Take input in matrices from a file to ease the process.

Solution:

```
int main() {
	const int ROWS1 = 2, COLS1 = 3, ROWS2 = 3, COLS2 = 2;

	int matrix1[ROWS1][COLS1];
	int matrix2[ROWS2][COLS2];
	int result[ROWS1][COLS2];

	ifstream fin("mat_mul1.txt");
	for (int i = 0; i < ROWS1; ++i)
		for (int j = 0; j < COLS1; ++j)
			fin >> matrix1[i][j];
	fin.close();

	cout << "\n  Matrix 1\n";
	for (int i = 0; i < ROWS1; ++i) {
		for (int j = 0; j < COLS1; ++j)
			cout << matrix1[i][j] << " ";
		cout << endl;
	}

	fin.open("mat_mul2.txt");
	for (int i = 0; i < ROWS2; ++i)
		for (int j = 0; j < COLS2; ++j)
			fin >> matrix2[i][j];
	fin.close();

	cout << "\n  Matrix 2\n";
	for (int i = 0; i < ROWS2; ++i){
		for (int j = 0; j < COLS2; ++j)
			cout<< matrix2[i][j]<<" ";
		cout << endl;
	}

	for (int i = 0; i < ROWS1; ++i)
		for (int j = 0; j < COLS2; ++j) {
			result[i][j] = 0;
			for (int k = 0; k < COLS1; ++k)
				result[i][j]+=matrix1[i][k] * matrix2[k][j];

		}

	cout << "\n  Result\n";
	for (int i = 0; i < ROWS1; ++i) {
		for (int j = 0; j < COLS2; ++j)
			cout << result[i][j] << " ";
		cout << endl;
	}
```

```
	return 0;
}
```

Exercise 8.4 (A)

Given an M X N matrix, find a 3 X 3 window in that matrix whose sum is maximum. Assume that the matrix contains non-negative integers. Take input in matrices from a file to ease the process.

Solution:

```
int main() {
	const int ROWS = 5, COLS = 7;

	int matrix[ROWS][COLS];

	ifstream fin("mat_window.txt");
	for (int i = 0; i < ROWS; ++i)
		for (int j = 0; j < COLS; ++j)
			fin >> matrix[i][j];
	fin.close();

	cout << "\n  Matrix\n";
	for (int i = 0; i < ROWS; ++i) {
		for (int j = 0; j < COLS; ++j)
			cout << matrix[i][j] << " ";
		cout << endl;
	}

	int max = -1, indexI, indexJ;
	for (int i = 0; i < ROWS - 2; ++i) {
		for (int j = 0; j < COLS - 2; ++j) {
			int sum = 0;
			for (int a = 0; a < 3; ++a) {
				for (int b = 0; b < 3; ++b) {
					sum = sum + matrix[i + a][j + b];
				}
			}
			if (sum > max) {
				max = sum;
				indexI = i;
				indexJ = j;
```

```
                        }
                }
        }
        cout << "\nMaximum sum is " << max << " found at index (" << indexI << "," << indexJ << ") of the matrix\n";

        for (int a = 0; a < 3; ++a) {
                for (int b = 0; b < 3; ++b)
                        cout << matrix[indexI + a][indexJ + b] << " ";
                cout << endl;
        }

        return 0;
}
```

Exercise 8.4 (B)

Given an M X N matrix, find a K X K window in that matrix whose sum is maximum. Take input in matrices from a file to ease the process.

Solution:

```
int main() {
        const int ROWS = 5, COLS = 7;

        int matrix[ROWS][COLS];

        ifstream fin("mat_window.txt");
        for (int i = 0; i < ROWS; ++i)
                for (int j = 0; j < COLS; ++j)
                        fin >> matrix[i][j];
        fin.close();

        cout << "\n  Matrix\n";
        for (int i = 0; i < ROWS; ++i) {
                for (int j = 0; j < COLS; ++j)
                        cout << matrix[i][j] << " ";
                cout << endl;
        }
        int K;
        cout << "\nEnter K (i.e. window size): ";
        cin >> K;
```

```
	while (K > ROWS || K > COLS) {
		cout << "Invalid value. Enter K (i.e. window size) again: ";
		cin >> K;
	}

	int max = -1, indexI, indexJ;
	for (int i = 0; i < ROWS - (K-1); ++i) {
		for (int j = 0; j < COLS - (K-1); ++j) {
			int sum = 0;
			for (int a = 0; a < K; ++a) {
				for (int b = 0; b < K; ++b) {
					sum = sum + matrix[i + a][j + b];
				}
			}
			if (sum > max) {
				max = sum;
				indexI = i;
				indexJ = j;
			}
		}
	}

	cout << "\nMaximum sum is " << max << " found at index (" << indexI << "," << indexJ << ") of the matrix\n";

	for (int a = 0; a < K; ++a) {
		for (int b = 0; b < K; ++b)
			cout << matrix[indexI + a][indexJ + b] << " ";
		cout << endl;
	}

	return 0;
}
```

Exercise 8.5

Take a number N as input from the user. Then take N number of sentences as input, store them in a 2D array and then print all sentences.

Solution:

```
int main() {
	const int SIZE = 100;
	const int LENGTH_OF_SENTENCE = 100;
	char sentences[SIZE][LENGTH_OF_SENTENCE];

	int N;
	cout << "Enter N (number of sentences you wanna enter): ";
	cin >> N;
	cin.ignore();

	for (int i = 0; i < N; ++i) {
		cout << "Enter sentence# " << i + 1 << ": ";
		cin.getline(sentences[i], LENGTH_OF_SENTENCE);
	}
	cout << endl;
	for (int i = 0; i < N; ++i)
		cout << "Sentence# " << i + 1 << ": " << sentences[i] << endl;
	return 0;
}
```

9. Passing Arrays to Functions

Exercise 9.1

Write a C++ program containing the following three functions:

1. ***fillArray:*** fills an integer array by taking input from the user. The function keeps on taking the input until the user enters -1.
2. ***printArray:*** prints an array
3. ***sumArray:*** returns the sum of all elements of the array

Solution:

```
void fillArray(int arr[], int& noOfElements) {
	noOfElements = 0;
	int input;
	cout << "Enter element# " << noOfElements + 1 << ": ";
	cin >> input;
	while (input != -1) {
		arr[noOfElements++] = input;
		cout << "Enter element# " << noOfElements + 1 << ": ";
```

```
                cin >> input;
        }
}

void printArray(const int arr[], int n) {
        cout << endl;
        for (int i = 0; i < n; ++i)
                cout << "Element# " << i + 1 << ": " << arr[i] << endl;
}

int sumArray(const int arr[], int n) {
        int sum = 0;
        for (int i = 0; i < n; ++i)
                sum += arr[i];
        return sum;
}

int main() {
        const int SIZE = 100;
        int arr[SIZE];

        int noOfElements;
        fillArray(arr, noOfElements);
        printArray(arr,noOfElements);
        cout << "Sum of the array is " << sumArray(arr, noOfElements) << endl;

        return 0;
}
```

Exercise 9.2

Write a C++ program containing the following four functions:

1. ***fillMatrix:*** fills a matrix (2D array) by taking input from the user
2. ***printMatrix:*** prints a matrix (2D array)
3. ***printRowWiseSum:*** prints a matrix (2D array) and also print sum of every row
4. ***findLargestElement:*** returns the largest element in the matrix (2D array)

Solution:

```
#define ROWS 2
#define COLS 3

void fillMatrix(int matrix[][COLS]) {
	for (int i = 0; i < ROWS; ++i) {
		for (int j = 0; j < COLS; ++j) {
			cout << "Enter element of row# " << i + 1 << " and col# " << j + 1 << ": ";
			cin >> matrix[i][j];
		}
	}
}

void printMatrix(const int matrix[][COLS]) {
	cout << endl;
	for (int i = 0; i < ROWS; ++i) {
		for (int j = 0; j < COLS; ++j) {
			cout << matrix[i][j] << " ";
		}
		cout << endl;
	}
}

void printRowWiseSum(const int matrix[][COLS]) {
	cout << endl;
	for (int i = 0; i < ROWS; ++i) {
		int sum = 0;
		for (int j = 0; j < COLS; ++j) {
			sum += matrix[i][j];
			cout << matrix[i][j] << " ";
		}
		cout << "Sum: " << sum;
		cout << endl;
	}
}

int findLargestElement(const int matrix[][COLS]) {
	int max=-1;
	for (int i = 0; i < ROWS; ++i)
		for (int j = 0; j < COLS; ++j)
			if (matrix[i][j] > max)
				max = matrix[i][j];

	return max;
}

int main() {
	int matrix[ROWS][COLS];
```

```
	fillMatrix(matrix);
	//printMatrix(matrix);
	printRowWiseSum(matrix);
	cout << "\nLargest element in the matrix is " << findLargestElement(matrix)<<endl;

	return 0;
}
```

Exercise 9.3 (Selection Sort)

```
void selectionSortDesc(int arr[], int n) {

	for (int i = 0; i < n - 1; ++i) {
		int maxIndex = i;
		for (int j = i + 1; j < n; ++j)
			if (arr[j] > arr[maxIndex])
				maxIndex = j;

		swap(arr[i], arr[maxIndex]);
	}
}
```

Exercise 9.4 (Bubble Sort)

```
void bubbleSort(int arr[], int n) {
	for (int i = 0; i < n - 1; ++i)
		for (int j = 0; j < n - i - 1; ++j)
			if (arr[j] > arr[j + 1])
				swap(arr[j], arr[j + 1]);
}
```

Exercise 9.5 (Binary Search)

Write a C++ function which receives an integer array, its size and a key as parameters and finds the key in the array. If the key is found, the function returns its index. Otherwise, it returns -1.

Solution:

```
int binarySearch(int key, int arr[], int n) {

	int left = 0, right = n - 1;
	while (left <= right) {
		int mid = (left+right)/2;

		if (key==arr[mid])
			return mid;

		if (key < arr[mid])
			right=mid-1;

		else
			left = mid + 1;
	}
	return -1;
}
```

10. Bitwise Operators

https://www.geeksforgeeks.org/bitwise-operators-in-c-cpp/#:~:text=The%20%7C%20(bitwise%20OR)%20in,every%20bit%20of%20two%20numbers

```
int main() {

	int A = 6;			//110
	int B = 3;			//011

	// Bitwise AND
	int bitwiseAND = A & B;
	cout << "A & B = " << A << " & " << B << " = " << bitwiseAND << endl;

	// Bitwise OR
	int bitwiseOR = A | B;
	cout << "A | B = " << A << " | " << B << " = " << bitwiseOR << endl;

	// Bitwise NOT
	cout << "  ~A  = " << " ~" << A << " = " << ~A << endl;
	//00000000000000000000000000000110
	//11111111111111111111111111111001  :  -7

	// Bitwise XOR
```

```
	int bitwiseXOR = A ^ B;
	cout << "A ^ B = " << A << " ^ " << B << " = " << bitwiseXOR << endl;

	// Shift right
	int M = 15;
	int n = 1;
	int shiftR = M >> n;
	cout << "M >> " << n << " = " << M << " >> " << n << " = " << shiftR << endl;

	// Shift left
	int P = 6;
	int q = 1;
	int shiftL = P << q;
	cout << "P << " << q << " = " << P << " << " << q << " = " << shiftL << endl;

	// Checking whether a number is even or odd
	int num = 7;
	if (num & 1)
		cout << num << " is an odd number\n";
	else
		cout << num << " is an even number\n";

	return 0;
}
```

11. Recursion

Exercise 11.1 (Counting up to N)

```
void printCounting(int n) {
	if (n == 0)
		return;

	printCounting(n - 1);
	cout << n << endl;
}

int main() {
	int n = 10;
	printCounting(n);
	return 0;
}
```

Exercise 11.2 (Backward counting from N)

```
void printBackwardCounting(int n) {
	if (n != 0){
		cout << n << endl;
		printBackwardCounting(n - 1);
	}
}

int main() {
	int n = 15;
	printBackwardCounting(n);
	return 0;
}
```

Exercise 11.3

Write a recursive function which prints a palindrome counting like this: 5 4 3 2 1 1 2 3 4 5

```
void printPalindromeCounting(int n) {
	if(n == 0)
		return;

	cout << n<<" " ;
	printPalindromeCounting(n - 1);
	cout << n <<" ";
}

int main() {
	int n = 7;
	printPalindromeCounting(n);
	return 0;
}
```

Exercise 11.4 (Factorial)

```
int fact(int n) {
	if (n == 0)
		return 1;

	return n * fact(n - 1);
```

```
}

int main() {
	int n = 6;
	cout<<n<<"! = "<<fact(n)<<endl;
	return 0;
}
```

Exercise 11.5 (Fibonacci Sequence)

```
int fib(int n) {

	if (n <=1)
		return n;

	return fib(n - 1) + fib(n - 2);
}

void printFibonacciSequence(int n) {
	if (n < 0) {
		cout << "Fibonacci Sequence is not defined for negative numbers.\n";
		return;
	}

	for (int i = 0; i <= n; ++i)
		cout << fib(i) << "  ";
}

int main() {

	int n = 12;

	//cout << "The term# " << n << " in Fibbonacci Sequence is " << fib(n) << endl;

	printFibonacciSequence(n);

	return 0;
}
```