

# National University of Computer and Emerging Sciences



## Programming Fundamentals CS188 Laboratory Manual

Course Instructor

Mirza Mubasher Baig

Lab Instructor(s)

Samia Akhter & Faraz Yousaf

Section

BDS-1A1 & A2

Semester


FALL 2021

**FAST School of Computing**

**Department of Software Engineering**

**FAST-NU, Lahore, Pakistan**

## National University of Computer and Emerging Sciences, Lahore Campus

	<b>Lab No 2</b>			
	Course Name:	Programming Fundamentals	Course Code:	CS 188
	Program:	BS(SE)	Semester:	Fall 2020
	Duration:	2.5 hours	Total Points:	20 + 20 + 20 20 + 20
	Lab Date:	Friday, October 1, 2021	Weight	3%
	Section:	BDS-1A	Page(s):	6

**Instruction/Notes:** Cheating during the lab will result in negative marks

**Topics Covered:** Computational problem solving and writing C++ programming using variables, expressions and conditional execution

The major focus of this lab is on Converting Algorithms into working C++ code and writing Algorithms using ALGORITHM 2.0

To write the working C++ code, you are required to use the online IDE available at available at <https://www.onlinegdb.com/>

### Activity 1: (Converting Algorithms into C++ Programs)

#### Problem No 1:

It is a known fact that sum of angles of a triangle is  $180^\circ$  and for any three values that add to 180 one can construct a triangle with angles equal to the given values.

The Algorithm given below can be used determine if a triangle can be created with angles equal to the values specified by the user. Use it to **create a C++ program that ask the user to specify three positive angles and then display a message stating if a triangle can be created using these angle or not.**

```
1  CR A, B, C
2  IN A, B, C
3  IF (A <= 0 OR B<=0 OR + C<=0) THEN
    OUT "ALL ANGLES MUST BE POSITIVE"
  ELSE
    IF (A + B + C == 180) THEN
      OUT "A Triangle can be created using these angles"
    ELSE
      OUT "A Triangle cannot be created using these angles"
    END IF
  END IF
4  OUT "THE END"
```

**Problem No 2:**

In the **Gregorian calendar** every year that is exactly divisible by four is a leap year, except for years that are exactly divisible by 100, but these centurial years are leap years if they are exactly divisible by 400. For example, the years 1700, 1800, and 1900 are not leap years, but the years 1600 and 2000 are. This logic can be easily converted into an Algorithm program that can determine if a year is a leap year or otherwise.

The following Algorithm has been written for this purpose and you are required to write a program using C++ to do the same.

```
1  CR Year
2  IN Year
3  IF ( Year MOD 400 == 0) THEN
    OUT Year, "IS A LEAP YEAR"
  ELSE
    IF ( Year MOD 100 == 0) THEN
      OUT Year, "IS NOT A LEAP YEAR"
    ELSE
      IF ( Year MOD 4 == 0) THEN
        OUT Year, "IS A LEAP YEAR"
      ELSE
        OUT Year, "IS NOT A LEAP YEAR"
      END IF
    END IF
  END IF
END IF
4  OUT "THE END"
```

### Problem No 3:

Digital computers represent different character of English language and digits from 0 to 9 using **numeric codes** (i.e. numbers). One possible way to represent such information is using **ASCII** (**A**merican **S**tandard **C**odes for **I**nformation **I**nterchange) codes. Following table shows the range of ASCII codes for various characters

Characters	ASCII Values
A – Z	65 – 90
a – z	97 – 122
0 – 9	48 – 57
special symbols	0 - 47, 58 - 64, 91 - 96, 123 – 127

In summary, the codes 65 to 90 are used for capital letters, codes 97 to 122 are used to represent small letters, codes 48 to 57 are used to represent numeric digits and all other codes below 128 are used to encode special characters.

The following pseudocode can be used to determine the type (i.e. capital, small, digit or special) of character coded by a given number. Convert this pseudocode into a working C++ program.

```
1  CR Code
2  IN Code
3  IF ( Code > 64 AND code < 91) THEN
    OUT Code , "Represents an upper case English alphabet"
  ELSE
    IF (Code > 96 AND code < 123) THEN
      OUT Code , "Represents a lower case English alphabet"
    ELSE
      IF (Code > 47 AND code < 58) THEN
        OUT Code, "Represents a digit"
      ELSE
        IF ( Code < 128) THEN
          OUT Code, "Represents a special character"
        ELSE
          OUT Code, "DOES NOT REPRESENTS AN ASCII Character"
        END IF
      END IF
    END IF
  END IF
END IF
4  OUT "THE END"
```

## Activity 2: (Writing Algorithms and Converting into C++ Programs)

### Problem No 1:

Write an Algorithm using ALGORITHM 2.0 and the corresponding C++ program that inputs a six-digit positive integer and prints the sum, and product of its six digits. Make sure that the number has to be a six digit number i.e. the numbers are in the range (100000 to 999999).

Sample Input: 153426

Sample output: SUM: 21, PRODUCT: 720

Sample Input: 15342

Sample output: NOT A SIX DIGIT NUMBER: TOO SMALL

Sample Input: 1534269

Sample output: NOT A SIX DIGIT NUMBER: TOO LARGE

**HINTS: DIV (C++ integer division i.e. /) and MOD (C++ integer mod i.e. %) operations might be useful for this purpose**

## **Problem No 2:**

Write an Algorithm using ALGORITHM 2.0 and the corresponding C++ program that takes x and y coordinates of a point **P** in 2D space and that of left-top and Right-bottom corner of a rectangle region. The program should be able to tell whether P lies inside the Rectangle or Not. Please remember that each point in 2D space has two coordinates (x, y)

### **Example Run 1**

#### **Sample Input:**

Enter X- Coordinate of Point P: 1

Enter Y- Coordinate of Point P: 1

Enter X- Coordinate of LEFT-TOP Corner P: 0

Enter Y- Coordinate of LEFT-TOP Corner P: 0

Enter X- Coordinate of RIGHT-BOTTOM Corner P: 2

Enter Y- Coordinate of RIGHT-BOTTOM Corner P: 2

**Sample Output: P lies inside the region**

### **Example Run 2**

#### **Sample Input:**

Enter X- Coordinate of Point P: 1

Enter Y- Coordinate of Point P: 4

Enter X- Coordinate of LEFT-TOP Corner P: 4

Enter Y- Coordinate of LEFT-TOP Corner P: 7

Enter X- Coordinate of RIGHT-BOTTOM Corner P: 6

Enter Y- Coordinate of RIGHT-BOTTOM Corner P: 10

**Sample Output: P lies outside the region**