**Compiler Construction Lab Mid**

Jahanzeb Razzaq

SP21-BCS-013

**QUESTION NO 1: Briefly describe the regex library of C#.**

**Ans:** The System.Text.RegularExpressions namespace in C# provides a powerful framework for working with regular expressions. Regular expressions (regex) are patterns used to match character combinations in strings.

**1. Regex Class:** This class represents an immutable regular expression. It provides methods for matching regular expressions against input strings, replacing matches, and splitting strings based on regex patterns.

**2. Match Class:** Represents the results of a single regular expression match. It provides properties and methods to access information about the match, such as the matched value, index, length, and groups.

**3. MatchCollection Class:** Represents a collection of Match objects. It is returned by methods like Regex.Matches() and allows iterating over multiple matches found in an input string.

**4. Group Class:** Represents a capturing group within a regular expression pattern. It provides properties to access the captured value, index, and length, as well as methods to retrieve captures within the group.

**5. Capture Class:** Represents a single captured sub string within a capturing group. It provides properties for the captured value, index, and length.

**6. RegexOptions Enumeration:** Provides options that modify the behavior of regex matching, such as case sensitivity, single-line mode, and ignore white space.

**Functionality of the Regex Class:**

The Regex class provides methods for working with regular expressions in C#. Some of the key methods include:

**IsMatch:**

Checks if a regular expression matches a specified input string.

**Match:**

Searches an input string for a substring that matches a regular expression pattern. Returns the first match found.

**Matches:**

Searches an input string for all occurrences of a substring that matches a regular expression pattern. Returns a collection of matches.

**Replace:**

Replaces all occurrences of a specified regular expression pattern with a specified replacement string.

**Split:**

Splits an input string into an array of substrings based on a regular expression pattern.

**Common Patterns:**

* \d: Matches any digit (0-9).
* \w: Matches any word character (alphanumeric + underscore).
* \s: Matches any whitespace character (spaces, tabs, line breaks).
* . Matches any character except a newline.

**Quantifiers:**

* \*: Matches 0 or more occurrences of the preceding pattern.
* +: Matches 1 or more occurrences of the preceding pattern.
* ?: Matches 0 or 1 occurrence of the preceding pattern.

**Character Classes:**

* [abc]: Matches any single character a, b, or c.
* [^abc]: Matches any single character except a, b, or c.
* [a-z]: Matches any lowercase letter.
* [0-9]: Matches any digit.

**Anchors:**

* ^ (Caret): Matches the start of a string.
* $ (Dollar Sign): Matches the end of a string.

**QUESTION NO 2:**

Make recursive descent or LL1 parser or recursive descent parser for the following grammar:

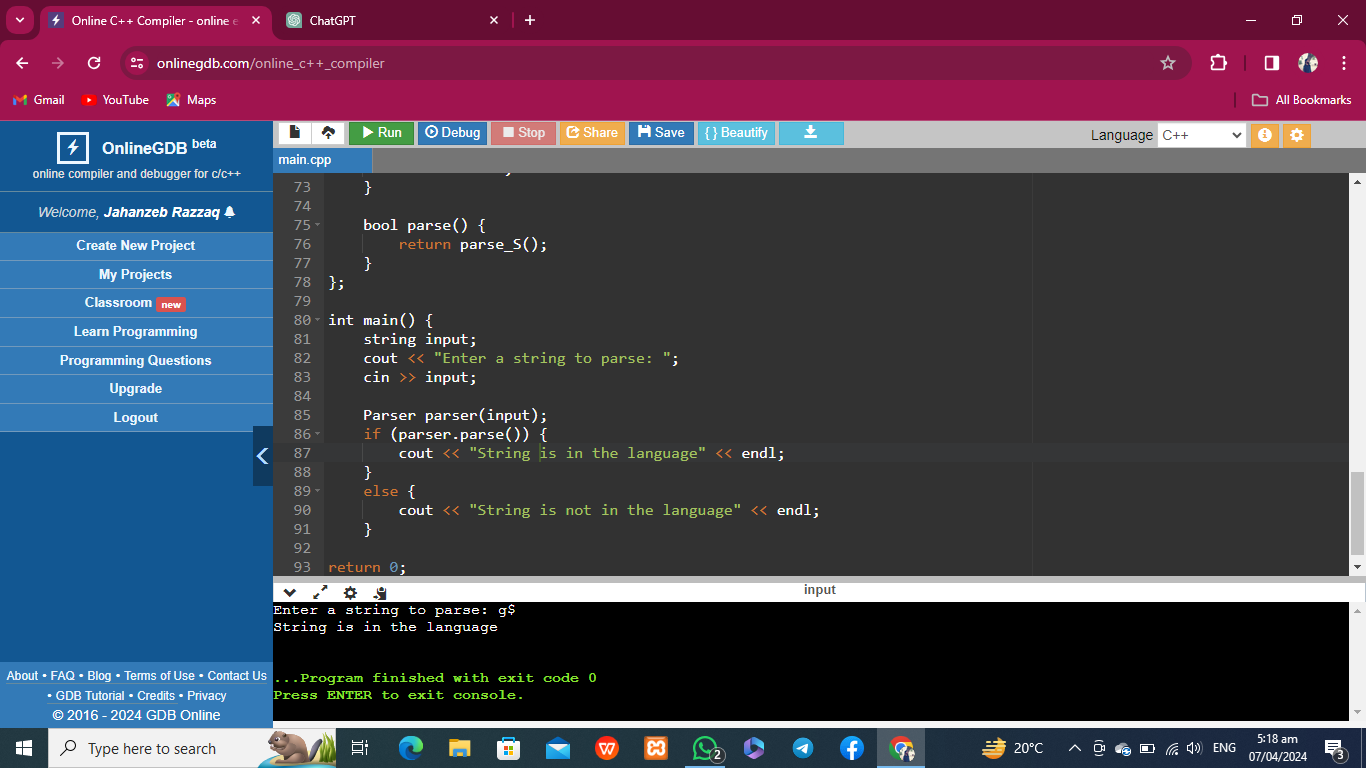
S -> X$

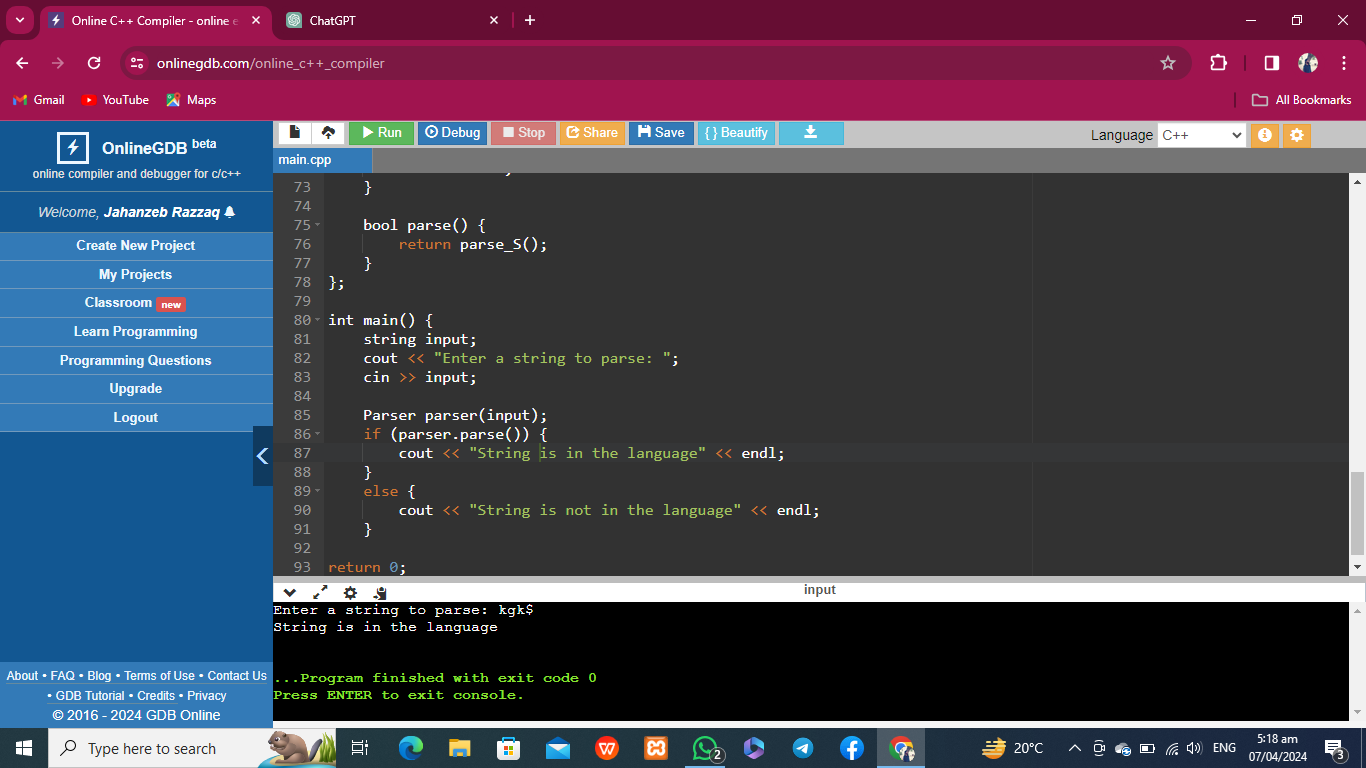
X -> X % Y |Y

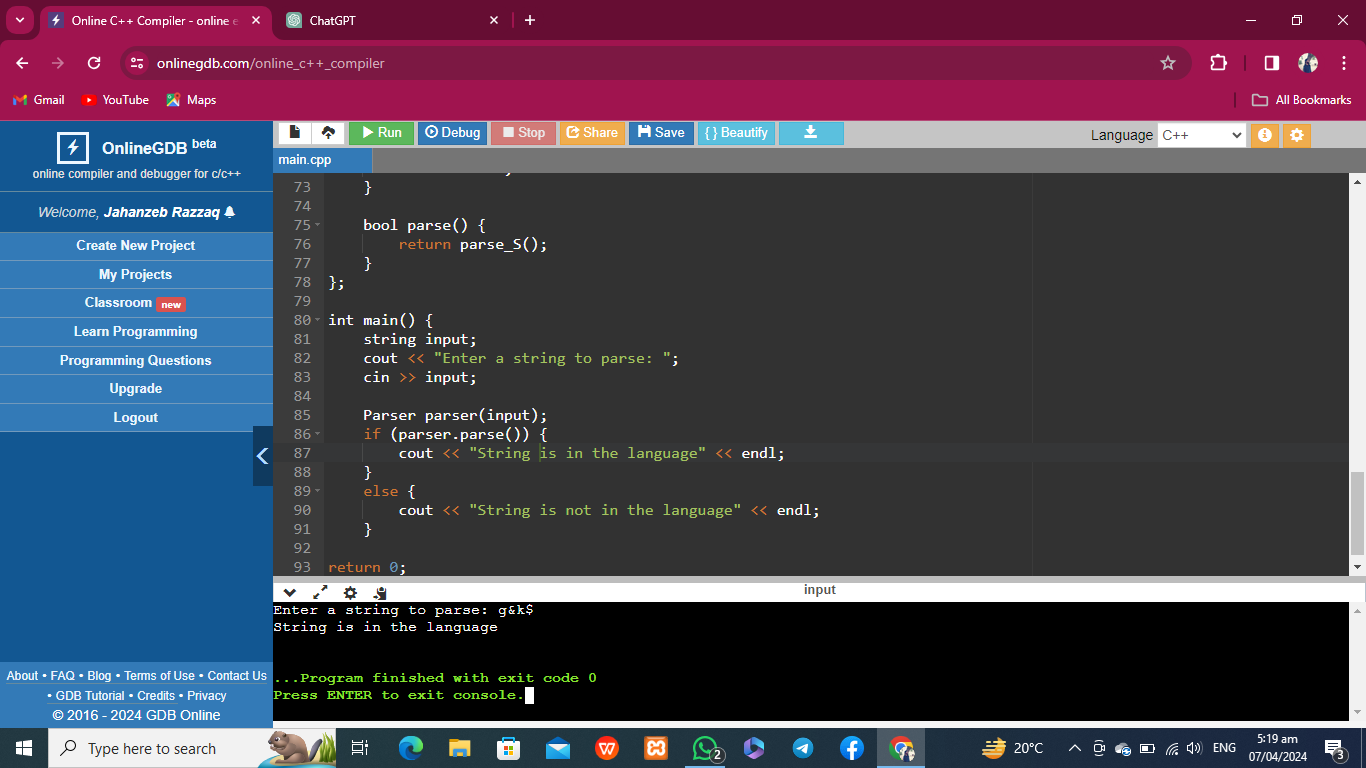
Y -> Y & Z |Z

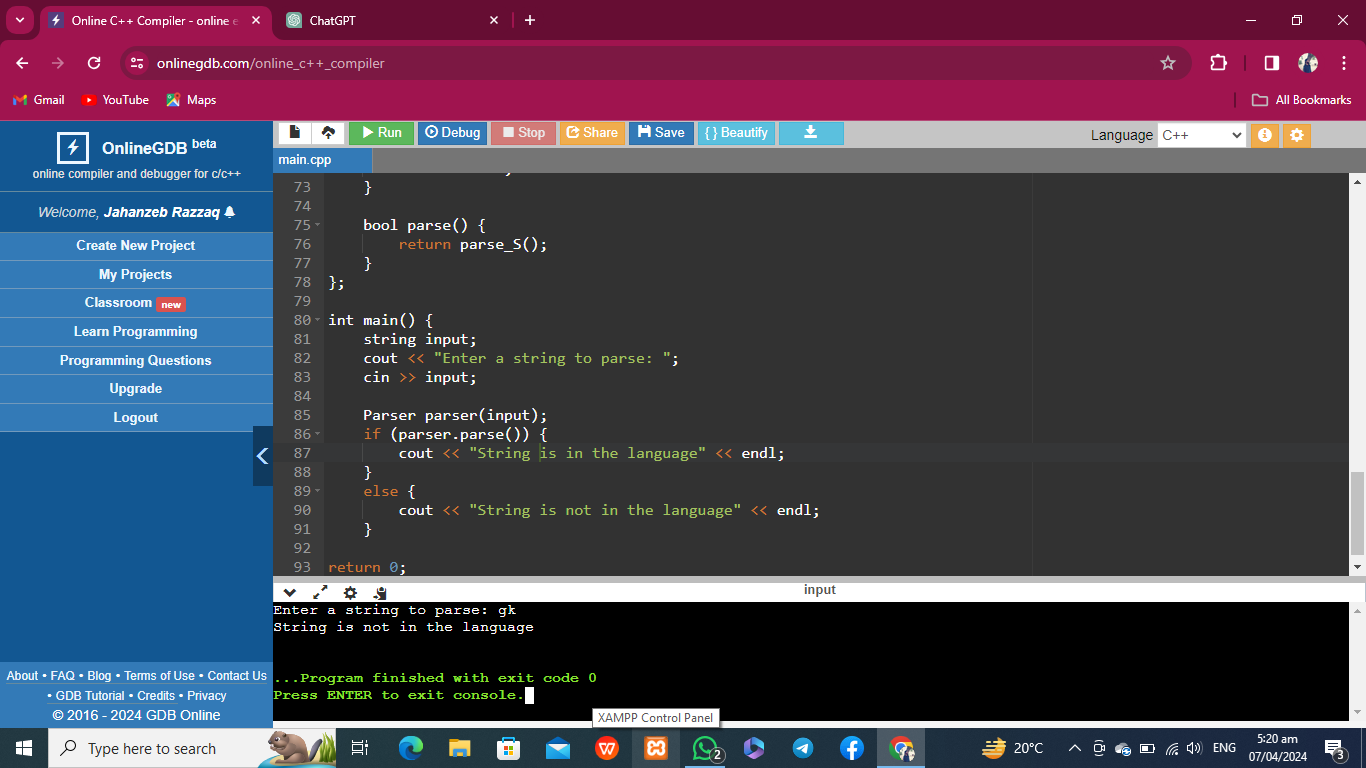
Z -> k X k | g

**ANS:**

****

****

****

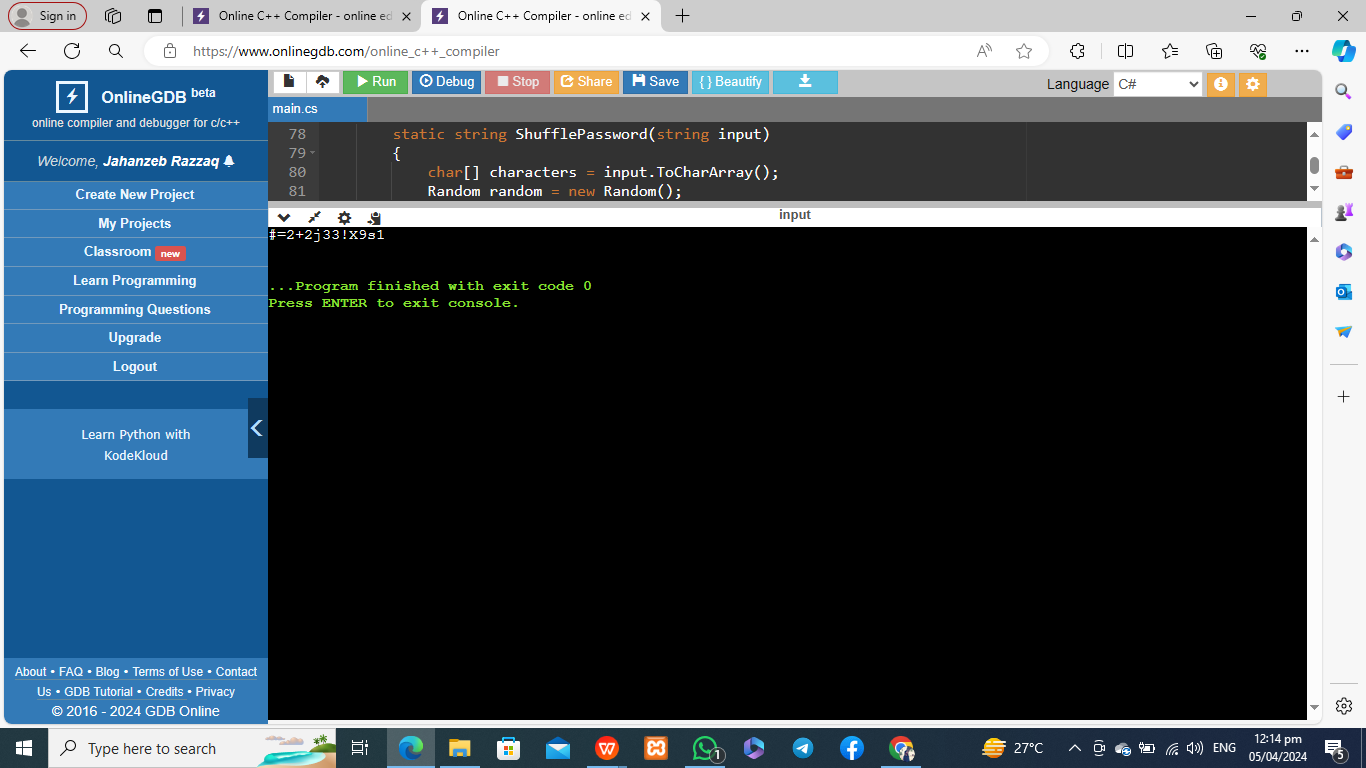
****

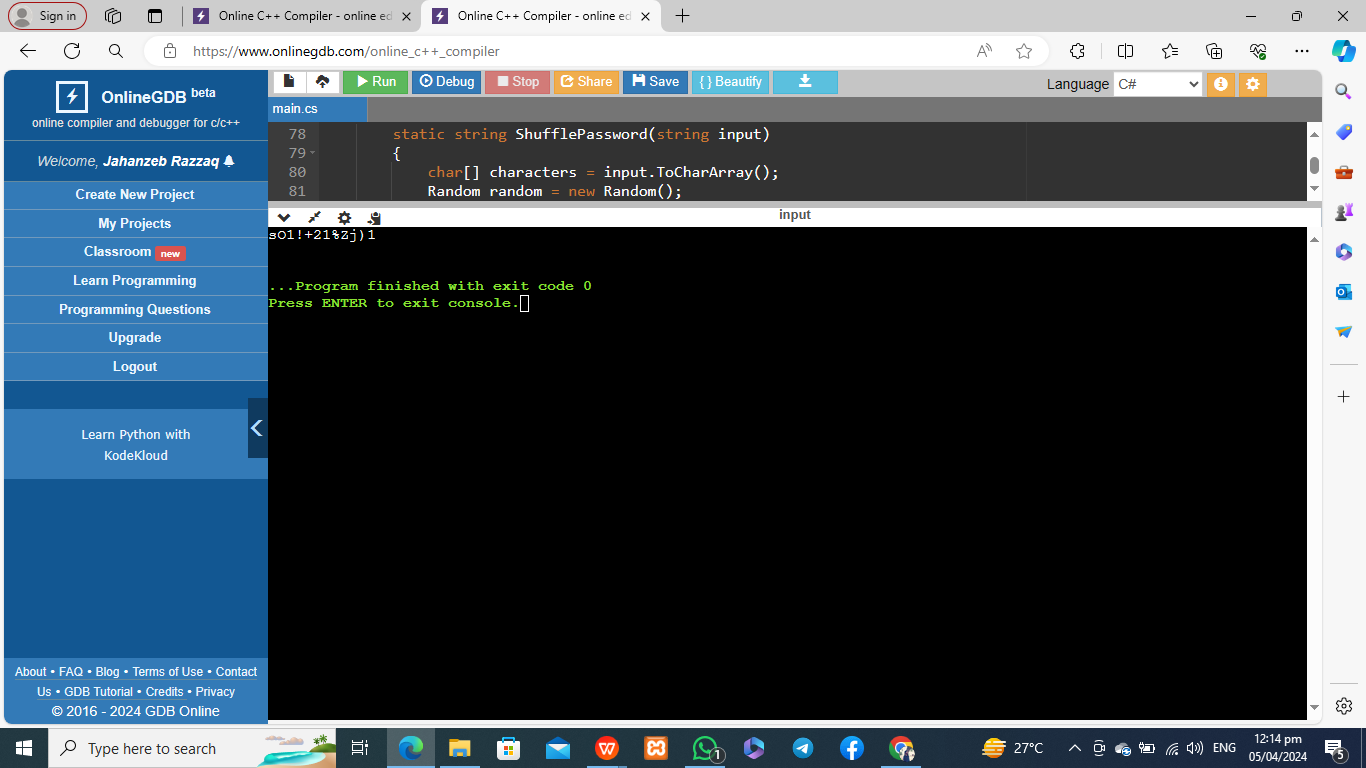
**QUESTION NO 3:**

Make a Password generator according the following rules:

1. Atleast one uppercase alphabet
2. Atleast 4 numbers , two numbers must be your registration numbers
3. Atleast 2 special characters
4. Must contain initials of first and last name
5. Must contain all odd letters of your first name.
6. Must contain all even letters of your last name.
7. maximum length of 16

**Ans:**

****

****