Canadian postal code

Canadian postal code in A1A 1A1 format.

```
^[A-CEGHJ-NPRSTVXY]\d[A-CEGHJ-NPRSTV-Z]\s?\d[A-CEGHJ-NPRSTV-Z]\d$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`[A-CEGHJ-NPRSTVXY]`	any of: A–C, “E”, “G”, “H”, J–N, “P”, “R”, “S”, “T”, “V”, “X”, “Y”
`\d`	any digit (0–9)
`[A-CEGHJ-NPRSTV-Z]`	any of: A–C, “E”, “G”, “H”, J–N, “P”, “R”, “S”, “T”, V–Z
`\s?`	optional (zero or one): whitespace
`\d`	any digit (0–9)
`[A-CEGHJ-NPRSTV-Z]`	any of: A–C, “E”, “G”, “H”, J–N, “P”, “R”, “S”, “T”, V–Z
`\d`	any digit (0–9)
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^[A-CEGHJ-NPRSTVXY]\d[A-CEGHJ-NPRSTV-Z]\s?\d[A-CEGHJ-NPRSTV-Z]\d$/;
re.test(value);
```

Python

```
import re
re.match(r"^[A-CEGHJ-NPRSTVXY]\d[A-CEGHJ-NPRSTV-Z]\s?\d[A-CEGHJ-NPRSTV-Z]\d$", value)
```

Java

```
Pattern.compile("^[A-CEGHJ-NPRSTVXY]\\d[A-CEGHJ-NPRSTV-Z]\\s?\\d[A-CEGHJ-NPRSTV-Z]\\d$")
.matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^[A-CEGHJ-NPRSTVXY]\d[A-CEGHJ-NPRSTV-Z]\s?\d[A-CEGHJ-NPRSTV-Z]\d
$");
```

Go

```
regexp.MustCompile(`^[A-CEGHJ-NPRSTVXY]\d[A-CEGHJ-NPRSTV-Z]\s?\d[A-CEGHJ-NPRSTV-Z]\d$`).
MatchString(value)
```

Ruby

```
/^[A-CEGHJ-NPRSTVXY]\d[A-CEGHJ-NPRSTV-Z]\s?\d[A-CEGHJ-NPRSTV-Z]\d$/.match?(value)
```

PHP

```
preg_match('~^[A-CEGHJ-NPRSTVXY]\d[A-CEGHJ-NPRSTV-Z]\s?\d[A-CEGHJ-NPRSTV-Z]\d$~',
$value);
```

Common pitfalls

- **Anchored to the whole string.** This pattern uses ^ and $, so it requires the entire input to match. To find it inside a longer text, drop the anchors and use the global (g) flag.
- **Escape it correctly per language.** In Java and JavaScript strings each backslash must be doubled (\\d); in Python, Go, and C# use raw/verbatim strings so the backslashes survive.
- **Validate beyond format.** Matching the format doesn't guarantee the value is real. Confirm the canadian postal code against a source of truth (database, API, or checksum) where it matters.