Please submit your code solution in a single zip file. You may write this in any programming language of your choice. To guide your solution these are the key ways we work:

1. We build distributed services and work with large scale data sync operations. So we are very keen that candidates understand time and space complexity.
2. As service teams we have end-to-end ownership of the delivery, so we are keen to see candidates put focus on testing e.g. adopting a TDD approach.
3. Given part of our delivery pipeline is code review, we would like to put some emphasis on maintainable code i.e. easily readable and easily understandable.
4. We would like to see proficiency of using the chosen language i.e use language features and built-in libraries rather than trying to reinvent the wheel.

This problem is a kind of code minimiser/compressor. It takes an input string and replaces identifiers with shorter ones. You will write a function that takes as input a string containing the source code and finds and replaces duplicate identifiers and returns the resulting string. You do not need to handle file input/output. For the purposes of this exercise an identifier is a string of letters (only). For example, "alice" is a single identifier while "jump4joy" is the identifier "jump", the non-identifier "4" and a second identifier "joy".

The second and subsequent times each identifier appears it is replaced by a dollar sign and a number which is the index of the first appearance of that identifier, counting the first identifier as 0, the next as 1, etc. Anything that is not an identifier is output as is and you do not need to parse the non-identifier parts. For example:

```
      minimise("you say yes, I say no you say stop and I say go go go")
=>
      "you say yes, I $1 no $0 $1 stop and $3 $1 go $12 $12"
```

As the example illustrates, this doesn't necessarily make things shorter as the single character identifier "I" in this example is replaced with two characters "$3". And it's not really a code minimiser either, because the result won't be valid code. As you will see in the larger example below, the replacements are made even inside comments and strings (which a real minimiser probably wouldn't do). That's because the code does not look at the non-identifier characters and doesn't know about quotes and comment markers. Whitespace is preserved because it treats space characters like any other non-identifier character: they're left unchanged.

If you believe there are problems with (or details left out of) the above requirements, please add a note describing the problems, and how you recommend resolving the problem.

Here's a larger example:

```
/*
 * Function to chop a string in half.
 */
public static string chop(string input) {
  if (input == null || input.isEmpty()) {
    return input;
  }
  if (input.length() % 2 == 1) {
    return "cannot chop an odd-length string in half";
  }
  return input.substring(input.length() / 2);
}
```

=>

```
/*
 * Function to chop a string in half.
 */
public static $4 $2($4 input) {
  if ($12 == null || $12.isEmpty()) {
    return $12;
  }
  $13 ($12.length() % 2 == 1) {
    $18 "cannot $2 an odd-$22 $4 $5 $6";
  }
  $18 $12.substring($12.$22() / 2);
}
```