Department of Computer Science

COS110 2016
Practical 5

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

1 Introduction

1.1 Submission

The submission deadline for this practical is **20 September 2016 at 11:59 PM**. You should aim to complete this assignment before the due date.

Fitchfork marks by comparing the output of your program with specified expected output on a line by line basis. For this reason you should pay close attention to the instructions for the output of your program. Also remember that names of files are case sensitive.

1.2 A Serious Warning

It is in your own interest that you, at all times, act responsible and ethically. As with any work done for the purpose of your university degree, remember that the University of Pretoria will not tolerate plagiarism. Do not copy a friend's assignment or allow a friend to copy yours. Doing so constitutes plagiarism, and apart from not gaining the experience intended, you may face disciplinary action as a result. For more information read the University of Pretoria's plagiarism policy on url http://www.library.up.ac.za/plagiarism/index.htm

1.3 Uploads

The student is advised to ensure that tarballs are created properly before uploading. The *.tar.gz* created by a student should not have any sub-folders. The zip folder should **ONLY** contain the source code.

Given code and data

Extract the content of the `Prac5.tar.gz` archive. After extracting this archive in a directory named `COS110/Prac5`, this directory should contain the files and directories shown in the following hierarchical structure:

```
COS110
└── Prac5
    ├── Prac5.tar.gz
    └── Task2
        └── CharString.h
```

Task 1: Exception Class [5]

An exception is a response to an event that happens during a program's execution. In this task, you are required to create a basic exception class that will be used in the following tasks. This class, **Exception**, will have the following:

- A member variable storing the reason for the exception as a string.
- A public function, what() , that returns the reason for the exception.
- A public constructor that takes in a string describing the reason for the exception and initialises the member variable.

Example usage:

```
try {
        throw Exception ("Test Exception");
} catch (Exception e) {
        cout « e.what() « endl;
}
```

Output:

```
Test Exception
```

There should be **no** output statement in your code, and **no** newline characters in the strings returned by *what()*.

On completion, create a tarball containing **Exception.h** and **Exception.cpp**. Upload it using the active fitchfork assignment called **Practical 5 - Exception Class**.

Task 2: CharString Class [20]

In ths task you will be required to implement your own string class, where you will use the Exception class created in Task 2 to indicate errors in various operator functions.

This class, **CharString**, needs to have the following operators overloaded:

- `The + operator`:
  This operator should return a CharString where the characters from the right hand side are appended onto the string on the left hand side.

  If the right hand string *is empty,* an exception should be thrown with the text *"The string is empty!"*.

  Example:

  "aaa" + *"bbb"* should return *"aaabbb"*

- `The - operator`:
  This operator should return a CharString where every instance of the character on the right hand side is removed from the string on the left hand side.

If there are no instances of the character in the string, then an exception should be thrown with the text *"No character '<character>' found!"* , where <*character*> is the character on the right hand side.

If the left hand string is empty, an exception should be thrown with the text *"The string is empty!"*.

If the result is an empty string, an exception should be thrown with the text *"The result is an empty string!"*.

Example:

"banana" - *"a"* should return *"bnn"*

- `The * operator`:
  This operator should return a CharString which consist of the left hand and right hand side interleaved. In other words, after every *n-th* character from the left hand string the *n-th* from the right string should follow.

  If the strings are not the same length, an exception should be thrown with the text *"Strings are unequal lengths!"*.

  Example:

  "aaa" * *"bbb"* should become *"ababab"*.

- `The / operator`

  This operator should return a CharString where every instance of every character from the right hand string should be removed from the left hand side.

  If there are no instances of any characters from the right hand side, then an exception should be thrown with the text *"No character found!"*.

  If either string is an empty string, an exception should be thrown with the text *"The string is empty!"*.

  If the result is an empty string, an exception should be thrown with the text *"The result is an empty string!"*.

  Example:

  "bananas" / *"and"* should become *"bs"*.

- `The == operator`:

  Should return true if every character from the left hand string is equal to the corresponding character from the right hand string. Otherwise, return false.

  If the strings are diffferent lengths, return false.

- `The [] operator`:

  The operator should return the character at index *i* of the string. If the index is out of bounds, an exception should be thrown with the text "Index out of bounds!".

- The ofstream operator, which should return (not output) the array in string representation.

- The assignment operator, which should make a deep copy of the string.

Additionally, each mathematical operator should have its corresponding assignment operator overloaded (`operator +=`, `operator -=`, `operator *= and operator /=`). These operators should modify the left hand parameter rather than returning the string hence the functions return type is *void*. For cases where the string increases in size (such as the `+ operator`) the size of the string should be dynamically increased.

CharString should also contain the following public methods:

- A copy constructor.
- A constructor which takes in a character array and a size, and initialises the CharString with the array.
- `int length()`, which returns the number of characters in the string.

A header file, **CharString.h**, is provided. **Do not edit any functions in this file**. You may add helper functions if you want. It is advised that you create a **main** file to test your implementation, and experiment with how to catch the exceptions.

There should be **no** output statements in your code, and **no** newline characters in any exception messages.

On completion, create a tarball containing **CharString.cpp** and **CharString.h**. Upload it using the active fitchfork assignment called **Practical 5 - CharString Class** .

Task 3: Exception Inheritance [10]

Inheritance and polymorphism can be used to simplify exception throwing. In this task, you will be required to extend the Exception class from Task 1 to make it simpler to throw exceptions in Task 2.

Rather than initializing the exception with a value, classes for each class of exception should be created:

- **EmptyExcepion**

  `Initialized with the string` *"The string is empty!"*.
- **OutOfBoundsExcepion**

  `Initialized with the string` *"Index out of bounds!"*.
- **UnequalSizeExcepion**

  `Initialized with the string` *"Strings are not the same size!"*.
- **EmptyResultExcepion**

  `Initialized with the string` *"The result is an empty string!"*.
- **CharacterNotFoundExcepion**

  It should be possible to use this exception in two ways:

1. If a character is passed into the constructor, the exception should contain the string *"No character '<character>' found!"* , where <*character*> is the character that was not found.
2. If no character is passed in, then the exception should contain the string *"No character '<character>' found!"* .

Example usage:

```
try {
        throw EmptyException;
} catch (Exception e) {
        cout « e.what() « endl;
}
```

Output:

```
The String is empty!
```

Modify your files from Task 2 to use these Exception classes, rather than passing in strings.

There should be no output statements in your code, and no newline characters in any strings.

Note: Implementations should be done in the header files.

On completion, create a tarball containing the **CharString.cpp**, **EmptyException.h**, **OutOfBoundsException.h**, **UnequalSizeException.h** and **CharacterNotFoundException.h**. Upload it using the active fitchfork assignment called **Practical 5 - Exception Inheritance**.