S Antoun

Colorado Mesa University

Computer Science and Engineering

CSCI250 Fall 2022

Practicum 3% bonus for the first 3 group solutions

Due at the end of your Week 10.

Background:

The aim of this exercise is to provide you with HEAPS of fun and program with CLASS.

Task 1 – Class 'File'

A text file is a collection of bytes. The following is an extract of a text file.

```
The cat sat on the mat
```

Your job in this lab is to write a simple class called `file`. The file class allows a text file to be associated with it at instantiation time through an initialization constructor. If no file is specified at instantiation time, then the default constructor is invoked and at a later time a function such as `associate` is executed. During the association process (either via the initialization constructor or `associate` method), the contents of the file is copied into a dynamically assigned buffer in one great big swoop. In order to do this, you will have to use appropriate function calls to work out how big the file is and then move the contents of the file into the buffer. You can assume the file is always going to be a text file. You can also assume that if an error occurs your program should terminate immediately. It should be noted that if the class instance is already holding data no other file can be associated with it – that is you can only call associate if no initialization constructor was invoked and it associate has never been called.

In addition to this, the class has two functions – one called `print` and the other `reverse`. The functions **print** the contents of the dynamically allocated buffer representing the file either in forward or reverse order. The buffer as represented IN the class is unchanged. That said each function takes an argument representing the number of characters (bytes) to skip printing after a character is printed.

For example given the above – the function `print(0)` should do the following;

```
The cat sat on the mat
```

If the function `print(1)` is invoked then 1 character is removed after a character is printed..

```
The cat sat on the mat
Tectsto h a
```

The reverse function does exactly the same thing in reverse. So the string above when displayed using reverse(0) looks like

```
tam ehe no tas tac ehT
```

When `reverse(1)` is invoked the string is printed out as:

```
a h otstceT
```

On first inspection the above may look incorrect but this is indeed correct because the last byte of the file is a newline.

When manipulating the internal string, you can not use array subscripts– you are to use pointer arithmetic. If there is no data to print, simply print out an error message.

Apart from these methods you are also to implement a `write` method which takes as an argument a string representing the name of the file to dump the contents of the class to i.e. whatever is associated with the buffer. This method should only generate a file if the class instance has data associated with it.. You can think of this as a very primitive copy program.

You also need to think carefully about what other things are needed in your class.

An example `main.cpp` and `file.h` have been provided to help you start. You are to fill in the gaps so that `main.cpp` will actually work. Code for the class should be placed in `file.cpp`. You are free to add extra code to `file.h` and `file.cpp` as needed.

Task 2 – Class 'File' – Miscellaneous methods (Optional)

In Unix environments there is a program called `cat` which produces a file concatenating various input files.

Lets say we have two file classes as illustrated by the code segment below:

```
file f1;
file f2;
```

How can we concatenate them together to produce a new file represented by class instance `f3`?

If `f1` after being associated with a file contains:

```
The ca
```

and `f2` after being associated with a file contains:

```
t sat on the mat
```

we could produce a resultant file by doing something like;

```
file f3;
f3.concatenate(f1);
```

```
f3.concatenate(f2);
```

Initially `f3` is empty but as you can see the contents of `f1` and `f2` are added to it. Once complete the contents of `f3` is now:

```
The cat sat on the mat
```

What if `f3` had data already associated with it? Well you would simply add to the end of the pre-existing data…

Make appropriate changes to your class so that this file concatenation is supported. It should be noted when `write` is invoked on `f3`, its entire contents is written to the nominated file. Make appropriate changes to `file.cpp` and `file.h`. You should test this with an appropriate main function.

Submission – Demo live in class

You are to demo the working and program code in the following files:

```
file.h
file.cpp
```