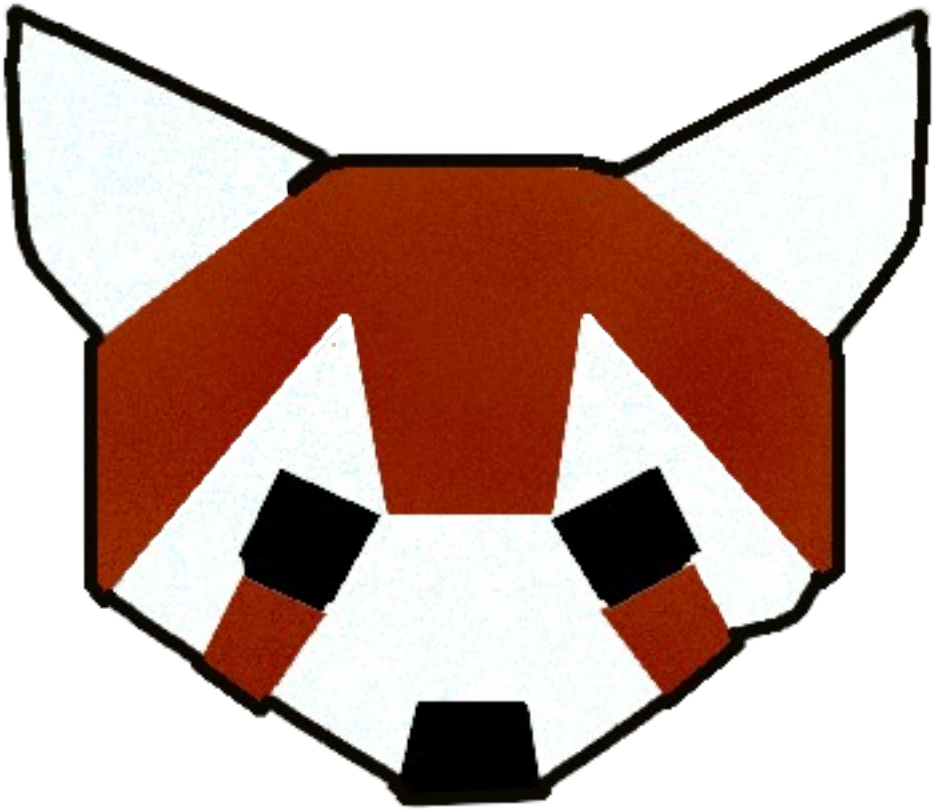
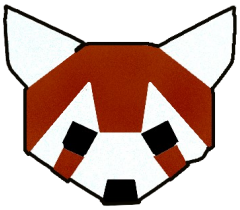


PANDALAB

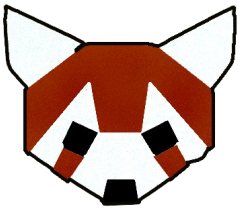
MYSTRING

TEMPLATE * THIS MIGHT BE BASIC !

- PandaLab -
pedagogie@ecole-89.com



INDEX



01 - Foreword

Your work must be returned through the **2022_My_String** git repository.

If you make a mistake and the folder you are using for your render is different, you will not be evaluated because you could not find your work.

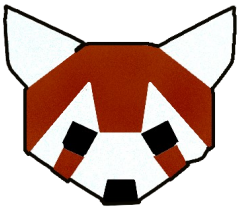
Your production must **strictly** respect all of the following rules:

- It must not contain **any** object file. (* .o)
- It must not contain **any** buffer files. (* ~, ###)
- It must not contain your final production (program or library)

Your rendering may still contain a main.cpp with which you have tested your work.
You must present a file **include, src,** a **Makefile,** a **README.MD** containing the user manual of your class and its functions, the **README** can be written in **English** or **French** at your **convenience**.

The readme must contain the first and last name of the author, as well as the year of production of the repository.

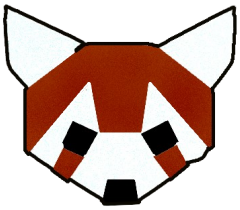
Be careful, when developing your code, any variable or function **that can be** constant **must be** constant, and **anything that can be** static **must be**.



02 – Authorized Functions / Libraries

Using a prohibited feature is considered cheating. Cheating causes the evaluation to stop and the medals to be lost.

For this work the use of the vector library, string is prohibited!
Use of c function to allocate memory is prohibited.



03 – MyBasicString

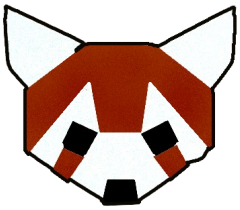
myBasicString.hpp

3:0 – Introduction:

Since the beginning of the year, we have been using the stl library.
With this came the **std :: string**, this class is used by many to store, parse, and print strings.
We are therefore going to embark on this work in order to produce our own string class.
Due to the number of functions that we will have to provide, this work is divided into several parts.
All the functions **that will follow** will be **obligatory for rendering**.
You can still add more functions, the most recent have not been requested, do not hesitate to implement them.

The **std :: string** is based on a **basic_string template**, Your job is to replicate this **basic_string template**, it's a **templated class**, and it should contain the ***CharacterType*** && ***Allocator*** parameters.

The default **allocator** should be the **allocator** of ***std :: allocator***, and allocate a ***CharacterType*** parameter.



3:1 – Constructor / Destructor:

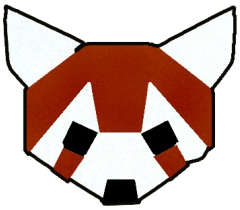
You must implement the following constructors and destructors:
For the center constructor, you need to fill the MyBasicString with the string str, for the others, manual.

```
MyBasicString(size_t l = 0)
MyBasicString(CharacterType *str)
~MyBasicString()
```

3:2 – Operator overload :

You must implement the following functions:

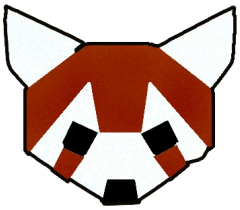
```
MyBasicString& operator= (const MyBasicString& str)
MyBasicString& operator= (const CharacterType* s)
MyBasicString& operator= (CharacterType c)
MyBasicString& operator= (MyBasicString&& str) noexcept
```



3:3 – Getter :

You must implement the following getter:

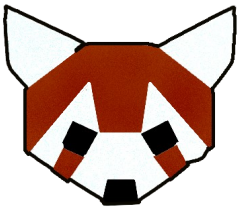
```
CharacterType& operator[] (size_t i)
const CharacterType& operator[] (size_t i) const
CharacterType& at (size_t pos)
const CharacterType& at (size_t pos) const
CharacterType& back()
const CharacterType& back() const
CharacterType& front()
const CharacterType& front() const
const CharacterType* c_str() const noexcept
const CharacterType* data() const noexcept
Allocator get_allocator() const noexcept
size_t      capacity()      const noexcept
size_t      size()          const noexcept
size_t      length()        const noexcept
size_t      max_size()      const noexcept
bool        empty()         const noexcept
```



3:4 – Learning :

You must implement the following functions:

```
MyBasicString& operator+= (const MyBasicString& str)
MyBasicString& operator+= (const CharacterType* s)
MyBasicString& operator+= (CharacterType c)
MyBasicString& append (const MyBasicString& str)
MyBasicString& append (const MyBasicString& str,
                       size_t subpos, size_t sublen)
MyBasicString& append (const CharacterType* s)
MyBasicString& append (const CharacterType* s, size_t n)
MyBasicString& append (size_t n, CharacterType c)
void push_back (CharacterType c)
void pop_back()
MyBasicString& insert (size_t pos, const MyBasicString& str)
MyBasicString& insert (size_t pos, const MyBasicString& str,
                       size_t subpos, size_t sublen)
MyBasicString& insert (size_t pos, const CharacterType* s)
MyBasicString& insert (size_t pos, const CharacterType* s, size_t n)
MyBasicString& insert (size_t pos,   size_t n, CharacterType c)
MyBasicString& replace (size_t pos, size_t len,
                        const MyBasicString& str,
                        size_t subpos,
                        size_t sublen)
MyBasicString& replace (size_t pos, size_t len, const CharacterType* s)
MyBasicString& replace (size_t pos, size_t len, const CharacterType* s,
size_t n)
MyBasicString& replace (size_t pos, size_t len,
                        const MyBasicString& str)
```



3:5 – Size Function :

You must implement the following functions:

```
void resize (size_t n)
void resize (size_t n, CharacterType c)
void shrink_to_fit()
void reserve (size_t n = 0)
```

3:6 – Clear :

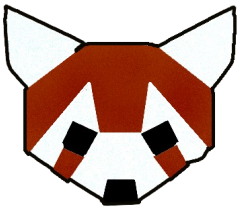
You must implement the following functions:

```
void          clear()            noexcept
MyBasicString& erase (size_t pos = 0, size_t len = npos)
```

3:7 – Special Operation :

You must implement the following functions:

```
void swap (MyBasicString& str)
CharacterType *substr (size_t pos = 0, size_t len = npos) const
size_t copy (CharacterType* s, size_t len, size_t pos = 0) const
int compare (const MyBasicString& str) const noexcept
int compare (size_t pos, size_t len, const MyBasicString& str) const
int compare (const CharacterType* s) const
int compare (size_t pos, size_t len, const char* s) const
```



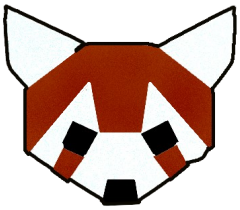
3:8 – Finder :

You must implement the following functions:

```
size_t find (const MyBasicString& str, size_t pos = 0) const noexcept
size_t find (const CharacterType* s, size_t pos = 0) const
size_t find (const CharacterType* s, size_t pos, size_t n) const
size_t find (CharacterType c, size_t pos = 0) const noexcept

size_t rfind (const MyBasicString& str, size_t pos = npos) const noexcept
size_t rfind (const CharacterType* s, size_t pos = npos) const
size_t rfind (const CharacterType* s, size_t pos, size_t n) const
size_t rfind (CharacterType c, size_t pos = npos) const noexcept
```

```
size_t find_last_of (const MyBasicString& str, size_t pos = npos) const noexcept
size_t find_last_of (const CharacterType* s, size_t pos = npos) const
size_t find_last_of (const CharacterType* s, size_t pos, size_t n) const
size_t find_last_of ( CharacterType c, size_t pos = npos) const noexcept

size_t find_first_of (const MyBasicString& str, size_t pos = 0) const noexcept
size_t find_first_of (const CharacterType* s, size_t pos = 0) const
size_t find_first_of (const CharacterType* s, size_t pos, size_t n) const
size_t find_first_of ( CharacterType c, size_t pos = 0) const noexcept
size_t find_first_not_of (const MyBasicString& str, size_t pos = 0) const noexcept
```

```
size_t find_first_not_of (const CharacterType* s, size_t pos = 0) const
size_t find_first_not_of (const CharacterType* s, size_t pos, size_t n) const
size_t find_first_not_of ( CharacterType c, size_t pos = 0) const noexcept
size_t find_last_not_of (const MyBasicString& str, size_t pos = npos) const noexcept

size_t find_last_not_of (const CharacterType* s, size_t pos = npos) const
size_t find_last_not_of (const CharacterType* s, size_t pos, size_t n) const
size_t find_last_not_of (CharacterType c, size_t pos = npos) const noexcept
```



04 – MyString Declaration

myBasicString.hpp

4:0 – MyString :

You need to make sure that if you call the type MyString you get a version of your templated class that is chars typed.

4:1 – MyString16_t :

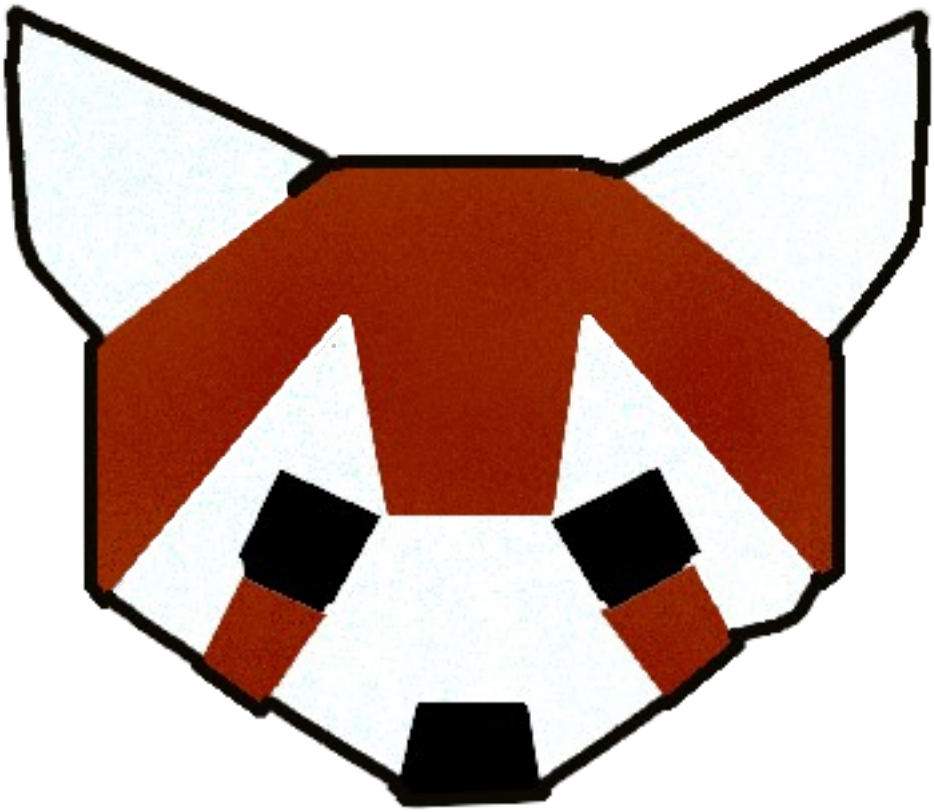
You need to make sure that if you call the type MyString16_t you get a version of your templated class that is chars16_t typed.

4:2 – MyString32_t :

You need to make sure that if you call the type MyString32_t you get a version of your templated class that is chars32_t typed.

4:3 – MyWString_t :

You need to make sure that if you call the type MyWString_t you get a version of your templated class that is wchars_t typed.



Little help, for those who have taken the courage to read everything!

- https://en.cppreference.com/w/cpp/string/basic_string
- https://en.cppreference.com/w/cpp/header/string
- https://en.cppreference.com/w/cpp/string
- https://en.cppreference.com/w/cpp/memory/allocator