**A User-Defined String Class**

For this computer assignment, you are to design and implement a user-defined String class to approximate the behavior of the string class from the STL. Your class should export the following methods:

• constructors

o String ( ): creates an empty String.

o String ( const String& s ): creates a copy of String s.

o String ( const String& s, const size\_t pos, const size\_t n = npos ): creates a copy of the portion of String s that begins at position pos and spans over n characters.

o String ( const char\* cs, const size\_t n ): creates a String from a copy of first n characters of C–string cs.

o String ( const char\* cs ): creates a String from a copy of characters of C–string cs.

o String ( const size\_t n, const char& c ): creates a String from a copy of n repetitions of character c.

• destructor

o ~String ( ): destroys the String object by deallocating all storage capacity allocated by String.

• assignment operator ( = )

o String& operator= ( const String& s ): assigns a copy of String s to a String.

o String& operator= ( const char\* cs ): assigns a copy of C–string cs to a String.

o String& operator= ( const char& c ): assigns a copy of character c ( as a C–string) to a String.

• operator+=

o String& operator+= ( const String& s ): appends a copy of String s to a String.

o String& operator+= ( const char\* cs ): appends a copy of C–string cs to a String.

o String& operator+= ( const char& c ): appends a copy of character c to a String.

• length ( ), size ( ) and capacity ( ) functions

o size\_t length ( ) const: returns the size of a String.

o size\_t size ( ) const: also returns the size of a String.

o size\_t capacity ( ) const: returns number of characters that a String can hold.

• empty ( ) and clear ( ) functions

o bool empty ( ) const: tests whether a String empty.

o void clear ( ): erases all characters of a String.

• subscription operator [ ] and index function

o char& operator[ ] ( const size\_t& pos ): returns a reference to character at position pos of a String.

o const char& operator[ ] ( const size\_t pos ) const: const version of the subscription operator.

o char& at ( const size\_t pos ): returns a reference to character at pos of a String and performs a range check. The at ( ) function of the string class in the STL throws an out\_of\_range exception and aborts the program in case of a range error, but the at ( ) function of your String class should only print an error message on stderr and not abort the program.

o const char& at ( const size\_t pos ) const: const version of the index function.

• substr ( ) function

o String substr ( const size\_t& pos = 0, const size\_t& n = 0 ): returns a substring of a String, which starts at position pos and spans over n characters (or until the end of the String, whichever comes first. If pos equals to String length, the function returns an empty String. The substr ( ) function of the string class in the STL throws an out\_of\_range exception and aborts the program in case of a range error, but the substr ( ) function of your String class should only print an error message on stderr and not abort the program.

• c\_str ( ) and data ( ) functions

o const char\* c\_str ( ) const: generates a C–string from a String.

o const char\* data ( ) const: also does the same.

• push\_back ( ) function

o void push\_back ( const char& c ): appends a copy of the character c to a String, increasing its size by one. It calls the expandMem ( ) function to increase the String’s buffer size.

• stream insertion (<<) and extraction (>>) operators and getline ( ) function

o friend ostream& operator<< ( ostream& os, const String& s ): inserts the characters of String s in the ostream os.

o friend istream& operator >> ( istream& is, String& s ): extracts the characters from istream is and inserts them in String s. It calls getline ( ) function to read the characters from is and stops reading at newline character.

o friend istream& getline ( istream& is, String& s, const char& del = ‘\n’ ): reads characters from istream is and stops reading at delimiter character del.

• operator+

o friend String operator+ ( const String& s1, const String& s2 ): returns a newly constructed String with its value being the concatenation of characters in String s1 followed by those in String s2.

o friend String operator+ ( const String& s, const char\* cs ): returns a newly constructed String with its value being the concatenation of characters in String s followed by those in C–string cs.

o friend String operator+ ( const char\* cs, const String& s ): This differs only from the previous one as the order of its two arguments are swapped.

o friend String operator+ ( const String& s, const char& c ): returns a newly constructed String with its value being the concatenation of characters in String s followed by character c.

o friend String operator+ ( const char& c, const String& s ): This differs only from the previous one as the order of its two arguments are swapped.

• relational operators

o friend bool operator== ( const String& s1, const String& s2 ): tests whether String s1 and String s2 are the same.

o friend bool operator!= ( const String& s1, const String& s2 ): tests whether String s1 and String s2 are different.

o friend bool operator< ( const String& s1, const String& s2 ): tests whether String s1 comes before String s2 in ASCII order.

o friend bool operator<= ( const String& s1, const String& s2 ): tests whether String s1 comes before String s2 in ASCII order or they are the same.

o friend bool operator> ( const String& s1, const String& s2 ): tests whether String s1 comes after String s2 in ASCII order.

o friend bool operator>= ( const String& s1, const String& s2 ): tests whether String s1 comes after String s2 in ASCII order or they are the same.

• expandMem ( ) function

o void expandMem ( const size\_t& n ): this is the *private* member function of the String class. If any of the functions of the String class needs to increase the buffer memory for a String, they call this function to reallocate the memory for size n.

Programming Notes

• Your code should work directly with your underlying representation and should make no calls to any of the methods in the STL string class.

• The npos is defined as a static value, and since it’s static, there is only one copy that is shared by all objects of the String class. The type of this value is size\_t, and for most systems, it is defined as unsigned int. You can assign –1 to npos. Then, it is guaranteed that npos will get the largest possible integer value in the system.

• If you implement one version of a function in the group of similar function, usually you can easily implement another function in the same group by calling the original function by different set of arguments. Sometimes, this really very easy. For example, when the result of the relational operator== is true, then the result of the relational operator!= is false.

The definition of the String class is given in the header file String.h, Put the implementation of the String class in your source file String.cc and include the full pathname of the header file String.h at the top of the source file.

To test your String class, a driver program prog9.cc is provided. There is also an input file prog9.d and the output file prog9.out for the correct output. All these files are in the same directory with String.h.