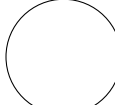
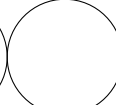
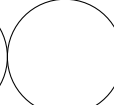
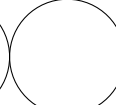



\$Id: cse111-2020q1-final.mm,v 1.67 2020-03-12 13:17:54-07 - - \$

page 1	page 2	page 3	page 4	Total / 40
				

Last Name :	
First Name :	
CruzID :	@ucsc.edu

No books ; No calculator ; No computer ; No email ; No internet ; No notes ; No phone. Neatness counts ! Points will be deducted for unreadable answers. Do your scratch work elsewhere and enter only your final answer on the exam.

For all questions, unless explicitly informed otherwise, assume all of the usual necessary `#include` directives are present followed by using `namespace std`;

1. Inheritance and virtual functions. All pointers in this question are to be **raw** pointers (**expr***). Code all function bodies inline. Make proper use of the keywords **virtual** and **override**.

- (a) Define an abstract base class called **expr**. [4✓]

- (i) **eval** has no arguments and returns a **double**.
- (ii) **print** returns **void** and has an **ostream** argument.
- (iii) **default** the default constructor and destructor.
- (iv) **delete** the copiers and the movers.

- (b) Class **leaf** inherits from **expr** and has a **double** value field. [2✓]

- (i) **eval** just returns this value and **print** just prints it.
- (ii) The constructor has one optional argument which defaults to 0.

- (c) Class **tree** inherits from **expr** and has two raw pointers to **exprs** (**left** and **right**).

- (i) **eval** returns the sum of the values of the two subtrees. [4✓]
- (ii) **print** prints the two subtrees recursively bounded by a pair of parentheses and connected by a plus signs. Example : `((6+8)+(9+44))`.
- (iii) The constructor takes two arguments (the left and right subtrees).
- (iv) The destructor deletes the subtrees.

2. Write a main function that reads in strings using `cin>>word` for each word, and at end of file, prints out all of the words in lexicographic order followed by a count of the number of times they occur. Use an appropriate data structure from the library. [2✓]

3. Define the template function `minmax`, whose arguments are a pair of forward iterators pointing at numbers of type `double`. Its result is a `pair<double, double>`, such that the `first` field is the minimum element and the `second` field is the maximum. Return `pair(NAN, NAN)` if the range is empty. (`NAN` is a macro in `<cmath>`). [3✓]

4. Define the template `operator<<` which will print a pair between braces (`{` and `}`) separated by a comma. Assume `operator<<` is already defined for the elements of the pair. [2✓]

For example, relating to the previous question :

```
vector<double> v1 {1.2, 8, 7, 0, -6, 9};
vector<double> ve;
cout << minmax(v1.begin(), v1.end()) << endl;
cout << minmax(ve.begin(), ve.end()) << endl;
would print :
{-6, 9}
{nan, nan}
```

5. Define the template `operator<<` whose second argument is a pair representing a range. It prints out the elements of the range with an open brace (`{`) at the beginning and a close brace at the end (`}`) and with a comma and space between each element. The iterators may point at anything for which `operator<<` is defined. [3✓]

For example :

```
vector<int> v {1,2,3,4,5};
cout << pair (v.begin(), v.end()) << endl;
will print :
{1, 2, 3, 4, 5}
```

6. Finish the function to draw a blue square using OpenGL functions. The x and y coordinates are the lower left corner of the square, and the length of one side is specified. The color should be specified as a static local variable. [3✓]

```
void draw_blue_square (GLfloat xpos, GLfloat ypos, GLfloat length) {
```

7. Given the following partial header file for an implementation of a vector of integers, write the code described in the following parts. *Invariant*: either: `data_` points at a raw array of integers and `size_` specifies the size; or: `data_` is `nullptr` and `size_` is 0. The specification is incomplete. Just make reasonable assumptions about what is left unspecified. All code for the following parts should be shown as it would appear in `intvec.cpp`.

```
class intvec {  
    private: int* data_ = nullptr;  
             size_t size_ = 0;  
    public:  intvec& operator= (const intvec&);  
            intvec& operator= (intvec&&);  
            ~intvec();
```

- (a) Code the copy operator=. [3✓]

- (b) Code the move operator=. [3✓]

- (c) Code the destructor. [1✓]

8. Write a complete function `readpipe`. It opens a pipe to read from the command given as its argument, and throws a `runtime_error` if the pipe can not be opened. If the `popen` is successful, it uses a buffer to read from the pipe, and each buffer read is appended to a `string`, which is then returned as the value of the function. [4✓]

```
string readpipe (const string& command) {  
    string result;  
    char buffer[1000];
```

9. Define the template function `merge`, with a return type of `void`. It has three template arguments: `InputIterator1`, `InputIterator2`, `OutputIterator`. It has five functional arguments: `begin` and `end` input iterators for the first input type, `begin` and `end` input iterators for the second input type, and a `begin` output iterator. Merge the two input ranges into the output range. Use `operator<` to compare elements, and assume the input ranges are already sorted into ascending order. [4✓]

10. Define `equal`, which takes two pairs of iterators and returns true if and only if the elements of the ranges given are equal, and the lengths of the ranges are equal. Assume only input iterators: You may not use `size()` or subtract iterators. Assume `operator==` is defined on the elements of the ranges. [2✓]

```
template <typename itor>  
bool equal (itor begin1, itor end1, itor begin2, itor end2) {
```