CSE 1325 Sample Final

Multiple Choice

1) A special class member that cleans up when an object is deleted is a(n)
   A) Friend
   B) Attribute
   C) Constructor
   D) Destructor ←

2) A single interface to multiple derived classes, enabling the same method call to invoke different derived methods to generate different results, is called
   A) Polymorphism ←
   B) Inheritance
   C) Encapulation
   D) Abstraction

3) Reuse and extension of fields and method implementations from another class is
   A) Polymorphism
   B) Inheritance ←
   C) Encapulation
   D) Abstraction

4) Bundling data and code into a container with restricted scope is
   A) Polymorphism
   B) Inheritance
   C) Encapulation ←
   D) Abstraction

5) In concurrent programs, critical regions of code are protected using
   A) A shared method
   B) A template
   C) A mutex ← Also known as a **mut**ual **ex**clusion object
   D) A setter

6) Which level of visibility makes class members accessible only within the same class?
A) Public
B) Protected
C) Private ← (or to friend functions)
D) Friend

7) A stream that is ready to accept or produce data has a stream state of
A) Good ←
B) Bad
C) Fail
D) Eof

8) In gtkmm, the layout (position) of widgets is controlled by
A) Their container, such as Gtk::VBox or Gtk::Grid ←
B) A layout manager class, such as Gtk::SpringLayout
C) Pixel-level specifications, such as Gtk::Button{200,300}
D) Cascading Style Sheet (css) documents

9) Which command will invoke Makefile rule "main"?
A) make
B) make main ← (“make” works only if it’s the first rule)
C) make rule main
D) make -r main

10) The statement "int i = 5;" stores the integer in:
A) Cache Memory
B) Heap Memory
C) Stack Memory ←
D) None of the above.

11) The implementation for C++ template Vector should be defined in
A) vector.h ←
B) vector.cpp
C) vector.tmpl
D) templates.cpp

12) In C++, generics (templates) may be defined for
A) Methods and functions
B) Functions and classes ←
C) Classes and enums
D) Enums and enum classes

13) Integer attribute "count" would be represented in a UML class as
A) int count
B) int : count
C) count int
D) count : int ←

14) In UML class diagrams, an open-head arrow (⇦) represents
A) Inheritance ←
B) Association
C) Composition
D) Aggregation

15) Which header must be included to use std::map?
A) iostream
B) map ←
C) sstream
D) algorithm

16) Which statement will sort std::vector<double> v?
A) std::sort(v.begin(), v.end()); ←
B) std::sort(v);
C) v.sort(v.begin(), v.end());
D) v.sort();

17) An object representing an error, propagated via special mechanisms until caught, is a(n)
A) Exception ←
B) Assertion
C) Operator
D) Instance

18) Which C++ expression will add value 3.14 to key "pi" for std::map<std::string, double> m?

A) m["pi"] = 3.14; ←
B) m.push_back(std::pair{"pi", 3.14});
C) m[m.size()+1] = std::pair("pi", 3.14);
D) m += std::pair{"pi", 3.14};

19) Which C++ expression is true if std::map m contains no pairs of data?
A) m.size() == 0
B) m.empty()
C) m.begin() == m.end()
D) All of the above ←

20) To stream out the number 3192 as a hex number with leading 0x (i.e., 0xc78), use
A) std::cout << 3192; // hex is the default format
B) std::cout << std::fullhex << 3192;
C) std::cout << std::hex << std::showbase << 3192; ←
D) std::cout(hex) << 3192;

21) For class Foo that is derived from class Bar, Foo's constructor can delegate to Bar's constructor using
A) Bar(int x);
B) Foo(int x) : Bar(x) { } ←
C) Foo(int x)->Bar(x);
D) Foo(int x).Bar(x);

22) A common response to a recurring problem that is usually ineffective and risks being highly counterproductive is
A) Class library
B) Polymorphic encapsulation
C) Software design pattern
D) Anti-pattern- ←

23) The int in std::vector<int> is an example of
A) a generic
B) a container
C) a collection
D) None of the above ← It is the template type

24) What keyword is needed to make a function polymorphic?
A) Void
B) Virtual ← (on the base class method)
C) Visual
D) All the above could work

25) std::cout is similar to which of these C functions?
A) scanf
B) sprintf
C) printf ←
D) None of the above

26) Which one of these streams will take input from a file?
A) ofstream
B) stringstream
C) cerr
D) ifstream ← which literally stands for “input file stream”

Free Response

1) Given an input file of people's names (which contain spaces), read in the file, store the values in a vector, display the values from the vector, sort the vector, write out the vector again and then write the sorted strings from the vector to a file.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
	ifstream MyIFile{"infile.txt"};
	ofstream MyOFile{"outfile.txt", ios::out};
	string FileLine;
	vector<string>Roster;

	if (MyIFile.is_open())
	{
		while (!MyIFile.eof())
		{
			getline(MyIFile, FileLine);
			Roster.push_back(FileLine);
		}
	}
	else
	{
		cout << "The file did not open" << endl;
		exit(0);
	}

	MyIFile.close();

	cout << "Roster before sorting" << endl;
	for (auto it : Roster)
		cout << it << endl;

	sort(Roster.begin(), Roster.end());

	cout << "\n\nRoster after sorting" << endl;
	for (auto it : Roster)
		cout << it << endl;

	cout << "\n\nWriting sorted Roster to file" << endl;
```

```
	for (auto it : Roster)
		MyOFile << it << endl;

	MyOFile.close();

	return 0;
}
```

2) Two RGB colors may be "mixed" by averaging their corresponding red, green, and blue values. Given the interface to class Color below, implement the constructor, operator+ method, and operator<< friend function such that main will print "Purple is (127,0,127)"

```
#include <iostream>
#include <ostream>

class Color {
  public:
   Color(int red, int green, int blue);
   Color operator+(const Color& rhs);
   friend std::ostream& operator<<(std::ostream& ost, const Color& c);
  private:
   int _red;
   int _green;
   int _blue;
};

int main() {
   Color red{255,0,0};
   Color blue{0,0,255};
   std::cout << "Purple is " << (red + blue) << std::endl;
}
```

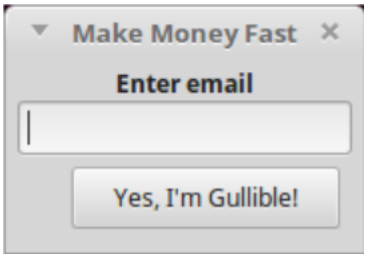
```
-------

Color::Color(int red, int green, int blue) : _red{red}, _green{green},
_blue{blue} { }

Color Color::operator+(const Color& rhs) {
    return Color{(_red+rhs._red)/2, (_green+rhs._green)/2,
(_blue+rhs._blue)/2};
}

std::ostream& operator<<(std::ostream& ost, const Color& c) {
    ost << '(' << c._red << ',' << c._green << ',' << c._blue << ')';
    return ost;
}
```

3) Given the following classes, add/correct/fill in the information necessary to have class Orangelo correctly inherit from class Orange and class Grapefruit. Class Orange and class Grapefruit are dervied from class Fruit. Class Fruit should be virtual. Create a vector named Basket that can hold an Orange, Grapefruit and Orangelo. Use a for loop to call member function whoamI for each element in the vector.

```
#include <vector>
#include <iostream>

class Fruit
{
    public:
           virtual void whoamI() = 0;
};

class Orange : public Fruit
{
    public:
           void whoamI()
           {
                 std::cout << "Orange";
           }
};

class Grapefruit : public Fruit
{
    public:
        void whoamI()
        {
            std::cout << "Grapefruit";
        }
};

class Orangelo : public Orange, public Grapefruit
{
    public:
        void whoamI()
        {
            std::cout << "Orangelo";
        }
};

int main(void)
{
      Vector<Fruit*> fruits;
      fruits.push_back(new Orange());
      fruits.push_back(new Grapefruit());
```

```
	fruits.push_back(new Orangelo());
	for(Fruit* f : fruits)
	{
		std::cout << f->whoamI() << std::endl;
	}

	return 0;
}
```

4) Fill in the blanks in the program below per the comments to create the dialog shown below.




```
#include <gtkmm.h>
#include <iostream>

int main(int argc, char *argv[])
{
    Gtk::Main kit(argc, argv);
    // (PART A) Allocate a Gtk::Dialog instance from the heap
    //   with the title bar set to "Make Money Fast"

Gtk::Dialog *dialog = new Gtk::Dialog{"Make Money Fast"};
        OR

   Gtk::Dialog *dialog = new Gtk::Dialog;
   dialog->set_title("Make Money Fast");

    // (PART B) Allocate a Gtk::Label instance from the heap
containing
    //   the message "Enter email" in bold text using Pango

Gtk::Label *label = new Gtk::Label("<b>Enter email</b>");

    label->set_use_markup(true);
    dialog->get_content_area()->pack_start(*label);

    // (PART C) Allocate a Gtk::Entry instance from the heap

Gtk::Entry *entry = new Gtk::Entry{};

    entry->set_max_length(50);
    dialog->get_content_area()->pack_start(*entry);

    // (PART D) Add a button to the dialog labeled "Yes, I'm
Gullible!"
    //     that returns 1 when clicked

dialog->add_button("Yes, I'm Gullible!", 1);
```

```
    // (PART E) Show all widgets in the dialog

dialog->show_all();

    // (PART F) Display the dialog, storing the id of the button
clicked
    //     in a newly instanced integer variable named result

      int result = dialog->run();

    dialog->close();
    while (Gtk::Main::events_pending())  Gtk::Main::iteration();

    // (PART G) If button "Yes, I'm Gullible!" was clicked, stream the
email address
    //     entered into the dialog to std::cout. Do nothing otherwise.

if (result == 1)
      std::cout << entry->get_text() << std::endl;

}
```

5) Consider the following code segment. Write a main function that creates two threads of some_function and then joins them back to the main function.

```
#include <iostream>
#include <thread>

void some_function(std::string message)
{
    std::cout << message << std::endl;
}
```

```
int main()
{
     thread t1(some_function, “Message 1”);
     thread t1(some_function, “Message 2”);

     t1.join();
     t2.join();

     return 0;
}
```

6) Given the following program, write the function maximum() using templates.

```
int main()
{
     int int1, int2, int3;
     double double1, double2, double3;
     char char1, char2, char3;

     // call maximum with int
     std::cout << "Input three integer values: ";
     std::cin >> int1 >> int2 >> int3;
     std::cout << "The max integer value is: " << maximum(int1,
int2, int3);

     // call maximum with double
     std::cout << "\n\nInput three double values: ";
     std::cin >> double1 >> double2 >> double3;
     std::cout << "The max double value is: " << maximum(double1,
double2, double3);

     // call maximum with char
     std::cout << "\n\nInput three characters: ";
     std::cin >> char1 >> char2 >> char3;
     std::cout << "The max char value is: " << maximum(char1,
char2, char3) << std::endl;

     return 0;
}
```

```
template <typename T>

T maximum(T value1, T value2, T value3)
{
   T maximumValue{value1}; // assume value1 is maximum

   if (value2 > maximumValue)
   {
      maximumValue = value2;
   }

   if (value3 > maximumValue)
   {
      maximumValue = value3;
   }

   return maximumValue;
}
```