CS 162 Exam II Winter 2023

Part I: True (A) / False (B), put A/B on the answer sheet (28 pts, 2 pts each)

1. Assuming elements are inserted into a queue in the order of 1 2 3 4 5, the order of removing those elements will be 1 2 3 4 5 as well.
2. The throw statement passes a value to the catch block, and it is legal to have a catch block with no parameter.
3. A recursive function can have at most one base case.
4. Recursive merge sort has a constant space complexity O(1).
5. A proper example of using recursion is when asking the user for an input until a valid one is provided.
6. In a derived class, you are not allowed to define new variables and member functions other than those that are in the base class.
7. If your program defines a class template, then the compiler will generate a class for each different data type for which it is instantiated.
8. Vector is an array that can grow and shrink in length while the program is running, where capacity refers to the number of elements in the vector.
9. All member functions in an abstract class need to be pure virtual.
10. A linked list is a linear data structure that supports random access to its elements.
11. Polymorphism is useful for creating more flexible and extensible code, since it allows for different behavior to be implemented based on the type of object being used.
12. Static binding occurs when the compiler binds a function call at compile time.
13. When deriving a class, you should only list functions in the base class that will be redefined.
14. If the member variables in a base class are marked as private, they are not inherited into its derived class.

Part II: Multiple Choices. Put your answers on the answer sheet (72 pts, 3 pts each)

15. Given the following function,
   ```c++
   int F (int a, int b) {
     if (b == 0) return a;
     return F(b, a % b);
   }
   ```
   What is the output of the following statement?
   ```c++
   cout << F(56, 24) << endl;
   ```
   A. 6
   B. 12
   C. 8
   D. No output, the program crashes
16. Given the following elements:
   \[6 \ 4 \ 8 \ 9 \ 2 \ 0 \ 1 \ 3\]
   To sort them in the ascending order, which sorting algorithm has been used based on the following intermediate steps?
   \[6 \ 4 \ 8 \ 9 \ 2 \ 0 \ 1 \ 3\]
   \[4 \ 6 \ 8 \ 9 \ 2 \ 0 \ 1 \ 3\]
   \[2 \ 4 \ 6 \ 8 \ 9 \ 0 \ 1 \ 3\]
   \[0 \ 2 \ 4 \ 6 \ 8 \ 9 \ 1 \ 3\]
   \[0 \ 1 \ 2 \ 4 \ 6 \ 8 \ 9 \ 3\]
   \[0 \ 1 \ 2 \ 3 \ 4 \ 6 \ 8 \ 9\]
   A. Merge Sort
   B. Insertion Sort
   C. Bubble Sort
   D. Selection Sort

17. What is the runtime complexity and space complexity of the sorting algorithm above?
   A. Runtime: O(1), Space: O(n)
   B. Runtime: O(n^2), Space: O(n)
   C. Runtime: O(n^2), Space: O(1)
   D. Runtime: O(n^2), Space: O(n^2)

18. What is the runtime complexity of the following code in terms of n? (Assuming n is greater than 0)
   ```java
   ...
   for (int i = 0; i < n; i++) {
       for (int j = 0; j < n; j*=2) {
           //...some O(1) operations here
       }
   }
   for (int i = 0; i < n; i++) {
       //...some O(1) operations here
   }
   ...
   ```
   A. O(n log n)
   B. O(n^3)
   C. O(n^2 log n)
   D. O(n log n + n)
19. What is wrong within the following code?

```cpp
int main () {
    vector <int> v;
    cout << v[0] << endl;
    return 0;
}
```

A. The program has a runtime error on `v[0]`, because the vector is empty.
B. The program has a runtime error on `vector<int> v`.
C. The program has a syntax error on `vector<int> v`.
D. The program has a syntax error on `v[0]`, because you can only use `v.at(0)` to access its element.

20. In Linked Lists, the head pointer `Linked_List* head`

A. is the first node in the list
B. points to the first node in the list
C. is always NULL
D. is undefined

21. What is an iterator in C++?

A. A type of container that stores data in a linked list.
B. A type of function that performs a specific task.
C. A way to access and manipulate the elements of a container.
D. A way to create new objects by combining existing ones.

22. Which of the following would correctly call the base class (Base) assignment operator overload from the derived class (Derived) assignment operator overload?

```cpp
Derived& Derived::operator= (const Derived& right){
    //what goes here?
}
```

A. `left = right;`
B. `this->right = Base.right;`
C. `Base::operator=(right);`
D. `Derived::right = Base::right;`

23. Using inheritance allows us to ________.

A. make our classes more modular
B. use polymorphism
C. eliminate duplicate code
D. All of the above

24. The ability to associate multiple meanings to one function name using dynamic binding is called ______.

A. inheritance
B. abstract
C. object-oriented
D. polymorphism
25. Which of the following is not true?
   A. An object of the base class may be stored in a variable of the derived class without casting.
   B. An object of the derived class may be stored in a variable of the base class.
   C. An object of a derived class that is derived from another class that is derived from a third class can be stored in a variable of the third class.

26. None of the above
   A catch block that expects an integer argument will catch
   A. All exceptions
   B. Any string exceptions
   C. All integer exceptions
   D. None of the above

27. Assuming both capacity and size of a vector are 0. First, add 16 elements to this vector using `push_back()`, then remove 8 elements using `pop_back()`. What is the capacity and size of the vector?
   A. Capacity: 0, size: 0
   B. Capacity: 16, size: 8
   C. Capacity: 32, size: 8
   D. Capacity: 8, size: 8

28. Which of the following are valid template prefixes?
   A. `template <class T>`
   B. `template <class V>`
   C. `template <class V, class T>`
   D. All of the above

For Questions 29-34, refer to the following three classes:

```cpp
class Fruit {
private:
    string name;
public:
    Fruit(const char* name) : name(string(name)) {} ~Fruit() { cout << "~Fruit()" << endl; } string get_name() { return name; };
    virtual string says() = 0;
};

class Apple : public Fruit {
public:
    Apple(const char* name) : Fruit(name) {} ~Apple() { cout << "~Apple()" << endl; } string says() { return "Sweet"; }
};

class HoneyCrisp : public Apple{
public:
```
HoneyCrisp (const char* name) : Apple(name) {}
~HoneyCrisp() { cout << "~HoneyCrisp()" << endl;}
string says() { return "Crunchy"; }
};

29. What would be printed by the following snippet?
HoneyCrisp h("honey_c");
cout << h.get_name() << " says " << h.says() << endl;

A. honey_c says
B. honey_c says Crunchy
C. honey_c says Sweet
D. This snippet would not compile, so nothing would be printed.

30. What would be printed by the following snippet?
Fruit* f = new Apple("apple");
cout << f->get_name() << " says " << f->says() << endl;

A. apple says
B. apple says Sweet
C. apple says Crunchy
D. This snippet would not compile, so nothing would be printed.

31. What would be printed by the following snippet?
Fruit* f = new Fruit("fruit");
cout << f->get_name() << " says " << f->says() << endl;

A. fruit says
B. fruit says Sweet
C. This snippet would compile, but it would not print anything.
D. This snippet would not compile, so nothing would be printed.

32. What is the output of the following snippet?
Fruit* f = new Apple("apple");
delete f;

A. ~Fruit()
B. ~Apple()
C. ~Apple()
   ~Fruit()
D. ~Fruit()
   ~Apple()
33. What term best describes the **Fruit** class?
   A. Virtual class
   B. Abstract class
   C. Pure virtual class
   D. None of the above

34. What term best describes the **Fruit** class's `says` function?
   A. Pure virtual function
   B. Virtual function
   C. Abstract function
   D. None of the above

For Questions 35-38, consider the template class below:

```cpp
template <class T>
class Array{
private:
   T* array;
   int size;
public:
   Array(int size);
   ~Array();
   T& at(int i);
};
```

35. What is the purpose of the template parameter `T`.
   A. It allows the class to be agnostic about the type of the array, which specified by `T`.
   B. It represents a specific class from the C++ Standard Template Library
   C. It's a C++ keyword that must be used for all template definitions
   D. It has no purpose

36. What would be the correct implementation of a constructor for the Array class that implemented enough space for an array of size elements of the template parameter type?
   A. `template <class T>
      Array<T>::Array(int size) : size(size) {
         this->array = new T[size];
      }`
   B. `Array::Array(int size) : size(size) {
      this->array = new T[size];
   }`
   C. `template <class T>
      Array<T>::Array(int size) : size(size) {
         this->array = new <T>[size];
      }`
   D. `Array::Array<T>(int size) : size(size) {
      this->array = new T[size];
   }`
37. What would be the proper way to use the **Array** class above to create an array of 10 integers?
   A. `Array<int> arr(10);`
   B. `Array<int, 10> arr;`
   C. `Array arr(10);`
   D. `Array arr(int, 10);`

38. How would you implement the function body of the destructor to properly free the memory allocated by the **Array** class?
   A. `delete array;`
   B. `delete [] array;`
   C. `delete <int> [] array;`
   D. `delete <T> [] array;`

**Extra Credit (6 pts, 2 pts each). Put your answers on the answer sheet:**

39. From the error message below, which line in which file causes the segmentation fault?
   ==11980== Process terminating with default action of signal 11 (SIGSEGV)
   ==11980==  Access not within mapped region at address 0x0
   ==11980==    at 0x401160: Maze::display_maze() const (maze.cpp:73)
   ==11980==    by 0x400DE5: main (main.cpp:17)
   A. Cannot be determined
   B. Line 17 in main.cpp
   C. Line 73 in maze.cpp
   D. B & C each generates a segmentation fault, so there are two segmentation faults in the program.

40. Which of the following is the correct GDB command to set a break point at line 17 in `main.cpp`?
   A. `break 17`
   B. `break main.cpp : 17`
   C. `break main 17`
   D. `break main -> 17`

41. Given the following two statements:
   ```
   int (*ptr1) [10];
   int *ptr2 [10];
   ```
   A. Both `ptr1` and `ptr2` are pointers to an array
   B. `ptr1` is a pointer to an array, `ptr2` points to an array of pointers
   C. Both `ptr1` and `ptr2` point to an array of pointers
   D. `ptr1` points to an array of pointers, `ptr2` is a pointer to an array