READ THIS NOW!

Failure to read and follow the instructions below may result in severe penalties.

- Print your name in the space provided below.
- Print your name and ID number on the Opscan form and code your ID number correctly on the Opscan form.
- Choose the single best answer for each question — some answers may be partially correct. If you mark more than one answer to a question, you will receive no credit for any of them.
- Unless a question involves determining whether given C++ code is syntactically correct, assume that it is. Unless a question specifically deals with compiler #include directives, you should assume the necessary header files have been included.
- Be careful to distinguish integer values from floating point values (containing a decimal point). In questions/answers that require a distinction between integer and real values, integers will be represented without a decimal point, whereas real values will have a decimal point, [1704 (integer), 1704.0 (floating point)].
- **This is a closed-book, closed-notes examination.**
- **No laptops, calculators or other electronic devices may be used during this examination.**
- **You may not discuss (in any form: written, verbal or electronic) the content of this examination with any student who has not taken it.**
- **You must return this test form when you complete the examination. Failure to adhere to any of these restrictions is an Honor Code violation.**
- There are 53 equal-valued questions. Each is worth 2 points.
- The answers you mark on the Opscan form will be considered your official answers unless the question directs otherwise.
- When you have finished, sign the pledge at the bottom of this page and turn in the test and your Opscan.

Do not start the test until instructed to do so!

Name (Last, First) _____

printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

_____

signature

Separate Compilation:

For the next few questions, consider the C++ program composed of two cpp files and two corresponding header files, as shown below.

//classes.h
class Guy {
public:
int height;
};
	//fun.h
	
	void Fn(Guy obj);

//main.cpp
#include "classes.h"
#include "fun.h"
 	
int main() {
Guy T;
Fn(T);
}
	//fun.cpp
	#include <iostream>
	using namespace std;
	#include "fun.h"
	
	void Fn(Guy my_guy) {
	cout << my_guy.height;
	}

1. Assume we want to compile (not build) main.cpp above. What will the compiler complain about when just main.cpp is compiled?

a.) Nothing
b.) Multiple definitions for Guy
c.) Undeclared identifier Guy
d.) Undeclared identifier Fn()
e.) both b and c
f.) both b and d
g.) both c and d
h.) both e and f

2.) If the organization shown above is used, what will the compiler complain about when fun.cpp is compiled?

a.) Nothing
b.) Multiple definitions for Guy
c.) Undeclared identifier Guy
d.) Undeclared identifier Fn()
e.) both b and c
f.) both b and d
g.) both c and d
h.) both e and f

3.) The order of the #include statements in main.cpp does not matter, (i.e. if #include "fun.h" is listed above #include "classes.h" the same compilation as above would result).

a.) True b.) **False**

4.) If we try to build the above program it gives an error. How can we fix the error?

a.) Use preprocessor directives #ifndef, #define, and #endif
b.) Add #include "classes.h" to the top of the file fun.h
c.) Add #include "fun.h" to the top of the file classes.h
d.) both a and b
e.) both b and c
f.) both a and c
g.) both d and e

5.) What is the purpose of the #include mechanism in C++?

a.) To "import" declarations of names that are declared elsewhere into a scope in which those names are to be used
b.) To "import" declarations of all names that are to be used into a scope
c.) To justify the inclusion of the #ifndef and #endif in the C++ language
d.) To create classes

6.) The effects of intelligent use of separate compilation include:

a.) Reduced compilation/link time for large projects after implementation changes in one function or class.
b.) Increased compilation/link time for large projects after implementation changes in one function or class.
c.) Easier re-use of independent code modules, such as data structures or data types
d.) Harder re-use of independent code modules, such as data structures or data types
e.) both a and c
f.) both a and d
g.) both b and c
h.) both b and d

Command Line Arguments:

7.) Suppose we have a dos command-line program named solcmd that accepts only one argument from the command line and suppose we type the following command at the dos prompt. (Assume "ifstream my_file;" has been initialized.)

solcmd *afilename.txt*

a.) my_file.open(argc[0]);
b.) my_file.open(argc[1]);
c.) my_file.open(argc[2]);
d.) my_file.open(argv[0]);
e.) my_file.open(argv[1]);
f.) my_file.open(argv[2]);
g.) None of the above because we need to use .c_str()

Class Basics

8.) If we have the classes A and B, and within the interface of B we have the statement "friend class A" then class B can access the private members of class A.

a.) True b.) **False**

9.) A static variable is initialized

a.) In the constructor, as in
b.) My_Class::My_Class():my_staticvar(0) {/*rest of constructor*/}
c.) At file scope at the beginning of the implementation
d.) In the class interface where it is declared

Assume the following class declaration for the next few problems:

```
class Electric_Heat {
private:
//true if heat is on
bool heat;
 //0-212 degrees
int temperature;

public:
Electric_Heat();
Electric_Heat(bool onoff,
              int setting);
void turnOn();
void turnOff();
void isOn();
void lower(int amount);
int getTemperature();
};
```

```
Electric_Heat::Electric_Heat() {
heat=false;//heat is off
temperature=73; //default
}

Electric_Heat::Electric_Heat(
      bool onoff, int setting) {
heat = onoff;
temperature=setting;
}

void Electric_Heat::turnOn(){
heat=true;
}
void Electric_Heat::turnOff() {
heat=false;
}
```

```
bool Electric_Heat::
                isOn() {
return heat;
}

void Electric_Heat::
                lower(int amount) {
//can be negative to increase
temperature-=amount;
}

int Electric_Heat::
                 getTemperature() {
return temperature;
}
```

10.) If we declare an array of type Electric_Heat of size 6, how many function or method calls will be made?

a.) 1
b.) 2
c.) 3
d.) **6**
e.) 12

11.) If we declare two Electric_Heat ers (Electric_Heat Kitchen(true, 70), Bedroom;), which of the following statements are valid?

a.) bool Electric_Heat_State= (Kitchen==Bedroom);
b.) Electric_Heat Room=Bedroom;
c.) Kitchen = Bedroom;
d.) Kitchen->lower(-10);
e.) cout << Bedroom;
f.) 2 of the above are valid
g.) 3 of the above are valid
h.) 4 of the above are valid
i.) all above are valid

12.) SHORT ANSWER: The statement(s) in question 11 that is/are valid raise(s) an issue if pointers are declared as data members of Electric_Heat. Explain.

The equal's operator…since it is a shallow copy, if there are pointers, both objects will have the same shared data since just the pointer is copied.

13.) SHORT ANSWER: Which of the methods in Electric_Heat should have been declared constant? Why?

isOn and getTemperature since they both do not change any data members.

14.) Say we have the following code in a separate cpp file. What happens?

```
#include <iostream>
using namespace std;

bool setOffAlarm(Electric_Heat Room) {
        return (room.temperature > 90);
}
void main () {
        Electric_Heat laundry(true, 85);
        laundry.lower(-15);
        if (setOffAlarm(laundry))
                cout << "WARNING!!";
}
```

a.) When run will display "WARNING"
b.) When run will not display ”WARNING"
c.) Causes a compilation error
d.) None of the above

15.) If Electric_Heat::isON() was implemented with the line "return (*this)->heat;" instead of the current implementation, what would happen?

a.) Compilation error
b.) Would work as before
c.) Returns a pointer to heat which would compile but breaks the encapsulation

16.) Which of the following **should always** be implemented by the programmer?

a.) Mutator methods
b.) Observer methods
c.) Constructors with parameters
d.) **Default constructor**
e.) all of the above

17.) Which of the following key words do not help set a class' access:

a.) public
b.) private
c**.) partial**
d.) protected

18.) Which of the following statements are true?

a.) Both the class and the struct have private members by default
b.) Both the class and the struct have public members by default
c.) The class has public and the struct has private members by default
d.) The class has private and the struct has public members by default

19.) What should a method never return?

a.) private data
b.) **a pointer to private data**
c.) void
d.) both a and b
e.) both b and c
f.) both a and c

Software Engineering:

20.) Which of the following is a defualt constructor?

a.) MyClass::MyClass(int x, int y);
b.) MyClass::MyClass();
c.) MyClass::MyClass(int x=1, int y=1);
d.) both a and b
e**.) both b and c**
f.) both a and c

21.) Which of the following declares two pointers to ints?

a.) int x, *y;
b.) int *x, y;
c.) int *x, *y;
d.) both b and c

22.) What is the denominator of the percentage of time that should be spent coding (mark on scantron).

6

23.) When developing a software system, the cost of correcting a "bug" depends on when it is caught. Which of the following are true?

a.) Errors caught during testing cost the least to fix
b.) Errors caught during design cost the least to fix
c.) Errors caught during coding cost the least to fix
d.) Errors caught during coding cost the most to fix

24.) Which of the following are NOT true: A software engineer spends time designing (instead of going straight to coding)

a.) So he can prevent errors from happening during coding,
b.) So he can communicate what needs to be coded to programmers,
c.) So maintainers of the software can make changes easier,
d.) So he can find missing requirements,
e.) So he can reuse earlier work, and
f.) So he can have a hard copy of the software.

25-31.) Put the following steps of the Waterflow Model in order.

a.) Requirements
b.) Testing
c.) Low Level Design
d.) Integration
e.) Specification
f.) High Level Design
g.) Coding

a,e,f,c,g,d,b

32.) Is program correctness verifiable?

a.) Yes
b.) No

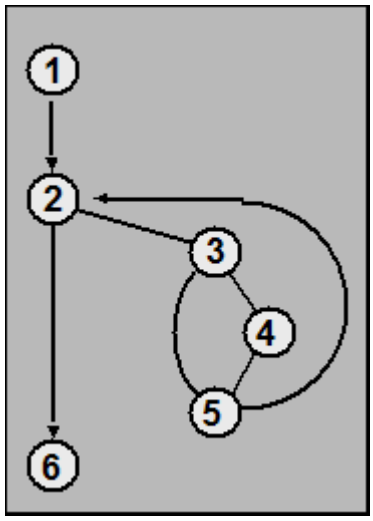
33.) The more errors you find during testing, the lower the probability that you will find more errors.

a.) True
b.) False

34-36) Lets say you are interested in doing Multiple Condition Coverage for a decision that had 7 conditions. How many test cases would you have to supply? (Enter each digit for questions 34-36. If the answer is less than 100, put a 0 for question 34. If the answer is less than 10, put a 0 for both question 34 and 35.)

2^7=128

37.) What is the cyclomatic complexity of the graph to the right?(Put answer on scantron.)
7-6+2=3



38.) Find as many independent paths as you can in the graph to the right.
1 2 6
1 2 3 5 2 6
1 2 3 4 5 2 6

Misc:

39.) The statements below are valid.

```
int x=5;
int y=&*x;
```

a.) True
b.) False

40.) A static variable has the same value for all objects of the same type.

a.) True
b.) False

Match the following:

41.) .cpp file **d**
42.) .h file **e**
43.) Accessor method **h**
44.) Data hiding **b**
45.) Destructor **c**
46.) Encapsulation **f**
47.) extern **g**
48.) Mutator method **i**
49.) Utility method **a**

a.) private method
b.) Abstract data type
c.) carries out "termination housekeeping"
d.) class implementation
e.) class interface
f.) class type
g.) global variables
h.) public const method
i.) public method

Match the following:

50.) Conditional compilation **c**
51.) Member-access specifier **a**
52.) Scope resolution operator **b**

a.) "." or "->"
b.) "::"
c.) #ifndef, #define, #endif, etc.

53.) We want to overload the > operator for the class MyClass. MyClass simply holds an integer x. Write the method implementation below.

```
bool MyClass::operator>(const MyClass& MC) const {
        return (x>MC.x);
}
```