**CS 162 Worksheet 6**

1. Accessor and Mutators:

Create a garage class that has a **dynamic array of vehicle structs**. Make sure you create an int variable to indicate the number of vehicles and follow the rules for encapsulation. Write the declarations for mutator, and accessor functions needed to access the members in the garage. Use const when necessary.

struct vehicle {

string name;

int num\_wheels, num\_seats;

bool motor;

};

class garage {

private:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; //dynamic array of vehicles

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; //number of vehicles

public:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; //accessor for dynamic array

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; //mutator for dynamic array

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; //accessor for number of vehicles

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; //mutator for number of vehicles

};

1. Use of const:

Given the following class declaration, explain each use of const. For a-d, tell what is legal, what is not illegal, and why.

class MyClass {

private:

int member1;

public:

void fun1(const int x);

int fun2() const;

};

1. void Myclass::fun1 (const int x){

int y = x;

}

1. void Myclass::fun1 (const int x){

x = member1;

}

1. int Myclass::fun2() const{

return this->member1;

}

1. int Myclass::fun2() const{

this->member1 = 2;

return this->member1;

}

3. Classes and objects:

Read and trace the code from the following three files, and answer the following questions.

|  |
| --- |
| garage.h: |
| garage.cpp:   |  |  | | --- | --- | |  |  | |  |  | |
| main.cpp: |

1. Between lines 11 and 12 in garage.h, which one is the default constructor, and which one is the non-default constructor?

1. What is printed by line 8 in main.cpp?

1. What is printed by line 9 in main.cpp?

1. Is anything printed by line 10 in main.cpp? If so, what?

1. Is anything printed by line 11 in main.cpp? If so, what?

1. What is printed by line 13 in main.cpp?

1. What is printed by line 14 in main.cpp?

1. Is anything printed by lines 15 and 16 in main.cpp? If so, what?
2. What is printed by line 17 in main.cpp?

1. Is anything printed by line 18 in main.cpp? If so, what?

1. Is anything printed by line 19 in main.cpp? If so, what?

1. Is anything printed by line 21 in main.cpp? If so, what?

1. Is anything printed by line 23 in main.cpp? If so, what?
2. What would happen if we additionally called g3.delete\_memory() at the end of main()?

4. Understanding errors

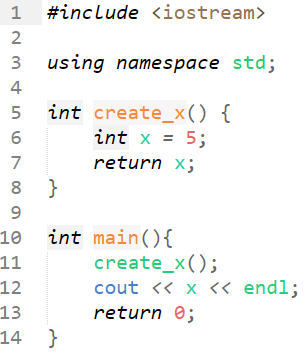
For each program and compiler / linker error shown below, answer the following questions: In what file and line of code does the error

appear? In your own words, what does the error mean? How would you fix this error?

Hint: compiler errors are described in great detail, and there are only a couple of common linker errors—you shouldn’t even need to look at

the code to understand, at least superficially, what’s causing the problem (though you may need to see the code to fully understand the

issue).

1. one.cpp:  
   

error:  
$ g++ one.cpp

one.cpp: In function ‘int main()’:

one.cpp:12:17: error: ‘x’ was not declared in this scope

12 | cout << x << endl;

| ^

1. two.cpp:  
   

error:

$ g++ two.cpp

two.cpp: In function ‘int main()’:

two.cpp:6:9: error: ‘my\_function’ was not declared in this scope

6 | my\_function();

| ^~~~~~~~~~~

|  |  |  |
| --- | --- | --- |
| three.h: | foo.cpp: | main.cpp: |

Error:  
$ g++ three.cpp three\_main.cpp

three\_main.cpp: In function ‘int main()’:

three\_main.cpp:4:13: error: too many arguments to function ‘void func()’

4 | func("Hello!");

| ~~~~^~~~~~~~~~

In file included from three\_main.cpp:1:

three.h:4:6: note: declared here

4 | void func();

| ^~~~

|  |  |  |
| --- | --- | --- |
| hello.h: | hello.cpp: | main.cpp: |

Error:  
$ g++ four.cpp four\_main.cpp

/bin/ld: /tmp/ccVopdR9.o: in function `main':

four\_main.cpp:(.text+0x5): undefined reference to `hello\_world()'

collect2: error: ld returned 1 exit status