COEN 79

Object-Oriented Programming and Advanced Data Structures

Assignment #2 - Due: April 15th, 2022 – 11:59PM

Name: Date:

- Number of questions: 10
- Points per question: 0.4
- Total: 4 points

1. What are the effects of using *inline member functions*?
   Write a class definition that includes at least two inline member functions.

- inline member function defined in class definition
  - you don't have to write implementation later
  - every time function is used, compiler will recompile the function definition and place it in code (saves execution time, inefficient space)

```
class car {
    private:
        int speed;
        string model;
    public:
       car {
          speed = 0;
          model = " ";

      int getSpeed() const {
            return speed
      }
      void zeroSpeed() {
          speed = 0;
      }
};
```

inline member functions

2. What are the differences between *references* and *pointers*?
(You can find many relevant articles online.)

- a pointer is a variable that holds the memory address of another variable
- a reference is an alias for an already existing variable and similarly implemented by storing address of a variable

INITIALIZATION:

```
int a = 10;
int *ptr = &a;
```

OR

```
int *ptr;
ptr = &a;
```

```
int a = 10;
int &ref = a;
```

← must be declared/defined on single line

- a pointer can be reassigned while a reference cannot
- pointer has its own memory address, reference shares with original variable
- multiple levels of pointers (ex. int ** dbl_ptr), can't for references
- use references for function parameters and return types
- use pointers for data structures (linked lists, trees) and if pointer arithmetic is needed

3. What is the *value semantics* of a class?
Write a class for which you can *rely on the compiler* to provide you with valid value semantics.

- value semantics of a class determine how values are copied from one object to another; assignment operator, copy constructor

```
class car {
    public:
        car();
        void setSpeed(int newSpeed) { speed = newSpeed; }
        int getYear() const { return year; }
    private:
        int speed;
        int year;
};
```

4. Where should we include the *invariants* of a class (header file or implementation file)? Please explain.

- in implementation file because only the programmer needs to know about the invariants of a class
  - critical in how the class is implemented but not used (by others)

5. The header of the `point` class is defined as follows:

```
1.  class point
2.  {
3.  public:
4.      // CONSTRUCTOR
5.      point (double initial_x = 0.0, double initial_y = 0.0);
6.
7.      // MODIFICATION MEMBER FUNCTIONS
8.      void set_x (double& value);
9.      void set_y (double& value);
10.
11.     // CONST MEMBER FUNCTIONS
12.     point operator+ (double& in) const;
13.
14. private:
15.     double x; // x coordinate of this point
16.     double y; // y coordinate of this point
17.
18. };
```

- Which line causes an error? Please explain why.

```
1.  main() {
2.      point myPoint1, myPoint2, myPoint3;
3.      double shift = 8.5;
4.      myPoint1 = shift + myPoint2;
5.      myPoint3 = myPoint1.operator+ (shift);
6.      myPoint1 = myPoint1 + shift;
7.  }
```

- What is the solution?

- line 4 in the main file causes an error
  - shift + mypoint2
    - shift: double
    - mypoint2: point

correct usage: (point) + (double)

↳ myPoint1 = myPoint2 + shift;

6. Why the following code does not compile? what is the solution?
Note: The notation "<>" is used for *template* classes. You will learn template classes in future chapters.

```
1.  #include < iostream >
2.  #include < list >
3.
4.  namespace coen79 {
5.      template < typename T >
6.      class list   {
7.          private:
8.              int array[20];
9.      };
10. }
11.
12. using namespace std;
13. using namespace coen79;
14.
15. int main(int argc, const char * argv[])
16. {
17.     using namespace std;
18.     list < int > v1;
19.     list < int > v2;
20.     return 0;
21. }
```

there is a naming conflict using list (in std and coen79)
specify intent with std::list or coen79::list

7. What is the output of this code? Discuss your answer.

```
1.  #include < iostream >
2.  using namespace std;
3.
4.  class CMyClass {
5.      public:
6.          static int m_i;
7.  };
8.
9.  int CMyClass::m_i = 0;
10.
11. CMyClass myObject1;
12. CMyClass myObject2;
13. CMyClass myObject3;
14.
15. int main() {
16.     CMyClass::m_i = 2;
```

```
17.     myObject1.m_i = 1;
18.
19.     cout << myObject1.m_i << endl;
20.     cout << myObject2.m_i << endl;
21.
22.     myObject2.m_i = 3;
23.     myObject3.m_i = 4;
24.
25.     cout << myObject1.m_i << endl;
26.     cout << myObject2.m_i << endl;
27. }
```

a static variable is shared between all instances of a class object thus the output is :

→ 1
→ 1 } changed line 17

→ 4
→ 4 } changed line 23

8. I wrote the following code to print 11 “Ouch!”. Why it is not working as expected?

```
1.  #include < iostream >
2.  using namespace std;
3.
4.  int main(int argc,  const char * argv[])
5.  {
6.      std::size_t i;
7.      for (i = 10; i >= 0; --i)
8.          cout << "Ouch!" << endl;
9.      return 0;
10. }
```

- size_t cannot be < 0 so the statement i >= 0 will always be valid and will cause an infinite loop

9. In the following code, indicate if the selected lines are legal or illegal:

```
#include <iostream>


class small
{
public:
    small( ) {size = 0;};
    void k() const;
    void h(int i);
    friend void f(small z);

private:
    int size;
};


void small::k() const
{
    small x, y;
    x = y; // LEGAL/ILLEGAL?
    x.size = y.size; // LEGAL/ILLEGAL?
    x.size = 3; // LEGAL/ILLEGAL?
};

void small::h(int i)
{

};

void f(small z)
{
    small x, y;
    x = y; // LEGAL/ILLEGAL?
    x.size = y.size; // LEGAL/ILLEGAL?
    x.size = 3; // LEGAL/ILLEGAL?
    x.h(42); // LEGAL/ILLEGAL?
};


int main() {

    small x, y;
    x = y; // LEGAL/ILLEGAL?
    x.size = y.size; // LEGAL/ILLEGAL?
    x.size = 3; // LEGAL/ILLEGAL?
    x.h(42); // LEGAL/ILLEGAL?

    return 0;
}
```

10. Modify the following code to generate the given output. Do not modify the `main` function.

```
1.  #include < iostream >
2.  using namespace std;
3.
4.  class box {
5.
6.  public:
7.      // Constructor definition
8.      box(double l = 2.0, double b = 2.0, double h = 2.0) {
9.              length  = l;
10.             breadth = b;
11.             height  = h;
12.             }
13.
14.     double volume() {
15.         return length * breadth * height;  }
16.
17. private:
18.     double length;   // Length of a box
19.     double breadth;  // Breadth of a box
20.     double height;   // Height of a box
21. };
22.
23. int main(void) {
24.     box Box1(3.3, 1.2, 1.5);   // Declare box1
25.
26.     box Box2(8.5, 6.0, 2.0);   // Declare box2
27.
28.     return 0;
29. }
```

Output:
Number of box objects created so far: 1
Number of box objects created so far: 2

- in public section of class add : static size_t boxCreated = 0;
- in constructor, increment boxCreated : ++boxCreated;
- also in constructor add :

cout << "number of box objects created so far: " << boxCreated << endl;