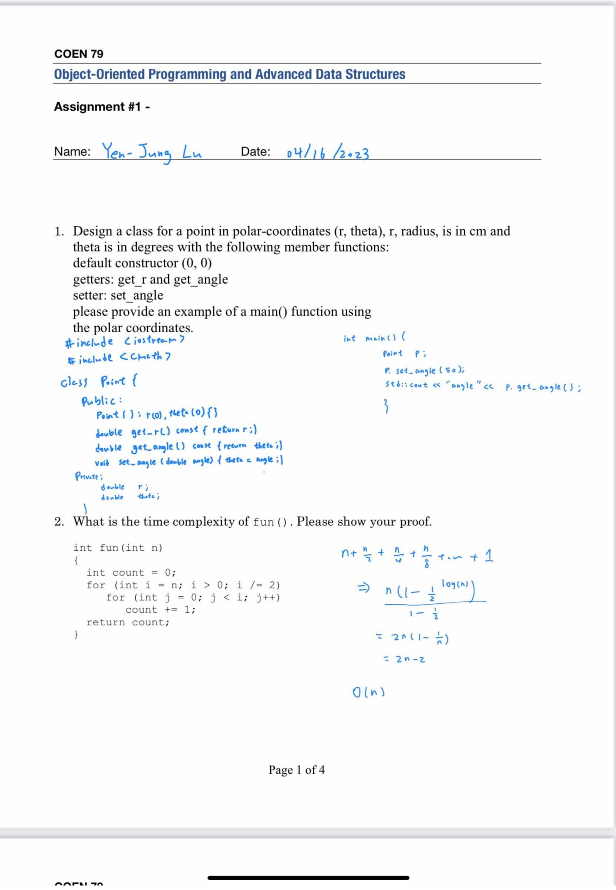
Assignment 2

Saturday, April 15, 2023 4:06 PM



COEN 79

Object-Oriented Programming and Advanced Data Structures

Assignment #1 -

Name: Yen-Jung Lu Date: 04/16/2023

1. Design a class for a point in polar-coordinates (r, theta), r, radius, is in cm and theta is in degrees with the following member functions:
   default constructor (0, 0)
   getters: get_r and get_angle
   setter: set_angle
   please provide an example of a main() function using
   the polar coordinates.

```
#include <iostream>
#include <Cmeth>

class Point {
   Public:
      Point ( ): r(0), theta (0){}
      double get_r() const { return r;}
      double get_angle () const {return theta;}
      void set_angle (double angle) { theta = angle;}
   Private;
         double r;
         double theta;
   }
```

```
int main() {
      Point P;
      P. set_angle (50);
      std::cout << "angle" << P. get_angle();
   }
```

2. What is the time complexity of `fun()`. Please show your proof.

```
int fun(int n)
{
  int count = 0;
  for (int i = n; i > 0; i /= 2)
     for (int j = 0; j < i; j++)
        count += 1;
  return count;
}
```

$n+\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+\dots+1$

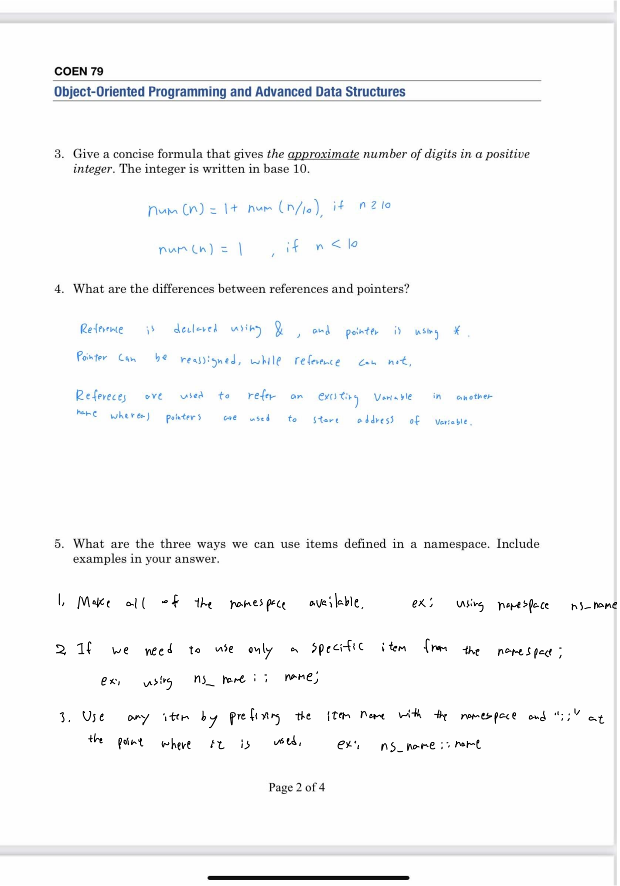
$\Rightarrow \frac{n(1-\frac{1}{2}^{\log(n)})}{1-\frac{1}{2}}$

$= 2n(1-\frac{1}{n})$

$= 2n-2$

$O(n)$

Page 1 of 4



COEN 79

Object-Oriented Programming and Advanced Data Structures

3. Give a concise formula that gives *the* *approximate* *number of digits in a positive integer*. The integer is written in base 10.

$num(n) = 1 + num(n/10)$, if $n \geq 10$

$num(n) = 1$, if $n < 10$

4. What are the differences between references and pointers?

Reference is declared using &, and pointer is using *.

Pointer can be reassigned, while reference can not,

Refereces are used to refer an existing Variable in another name whereas pointers are used to store address of Variable.

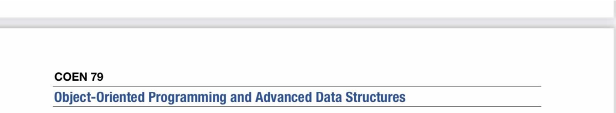
5. What are the three ways we can use items defined in a namespace. Include examples in your answer.

1, Make all of the namespace available. ex; using namespace ns_name;

2, If we need to use only a specific item from the namespace;
   ex, using ns_name ; ; name;

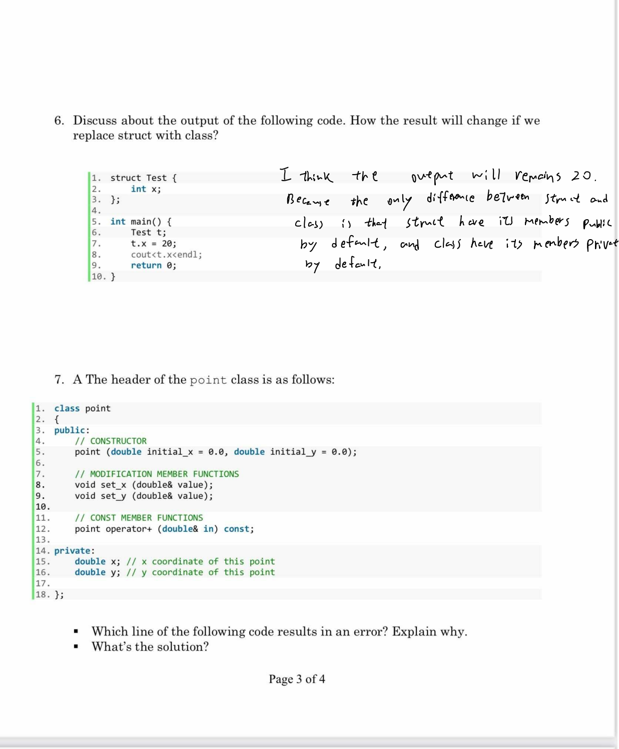
3. Use any item by prefixing the item name with the namespace and "::" at the point where it is used. ex; ns_name :: name

Page 2 of 4



COEN 79

Object-Oriented Programming and Advanced Data Structures



6. Discuss about the output of the following code. How the result will change if we replace struct with class?

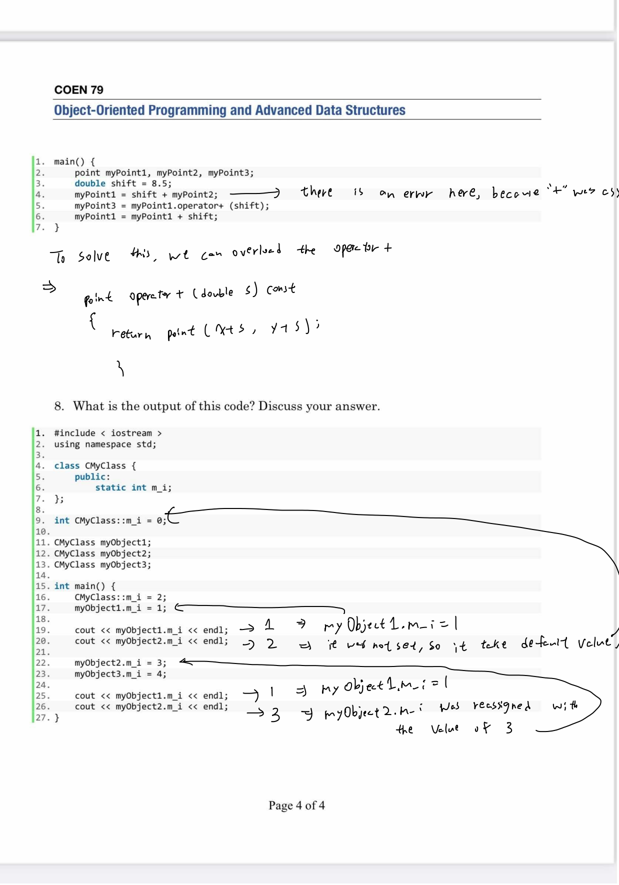
```cpp
struct Test {
    int x;
};

int main() {
    Test t;
    t.x = 20;
    cout<t.x<endl;
    return 0;
}
```

I think the output will remains 20. Because the only difference between struct and class is that struct have its members public by default, and class have its members private by default.

7. A The header of the point class is as follows:

```cpp
class point
{
public:
    // CONSTRUCTOR
    point (double initial_x = 0.0, double initial_y = 0.0);

    // MODIFICATION MEMBER FUNCTIONS
    void set_x (double& value);
    void set_y (double& value);

    // CONST MEMBER FUNCTIONS
    point operator+ (double& in) const;

private:
    double x; // x coordinate of this point
    double y; // y coordinate of this point

};
```

- Which line of the following code results in an error? Explain why.
- What's the solution?



```cpp
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. }
```

→ there is an error here, becaue "+" was assigned to take double, but it is trying to take point here.

To solve this, we can overload the operator +

⇒ point operator + (double s) const
{
return point (x+s, y+s);
}

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

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

class CMyClass {
    public:
        static int m_i;
};

int CMyClass::m_i = 0;

CMyClass myObject1;
CMyClass myObject2;
CMyClass myObject3;

int main() {
    CMyClass::m_i = 2;
    myObject1.m_i = 1;

    cout << myObject1.m_i << endl;
    cout << myObject2.m_i << endl;

    myObject2.m_i = 3;
    myObject3.m_i = 4;

    cout << myObject1.m_i << endl;
    cout << myObject2.m_i << endl;
}
```

→ 1 ⇒ myObject1.m_i = 1

→ 2 ⇒ it was not set, so it take default value, it is static, so it is 2

→ 1 ⇒ myObject1.m_i = 1

→ 3 ⇒ myObject2.m_i was reassigned with the value of 3