

### READ THE INSTRUCTIONS

- Use pencil only
- Write your name at the top of all pages turned in.
- Do not remove the staple from your test.
- Handwriting that is illegible (messy, small, not straight) will lose points.
- Indentation matters. Keep code aligned correctly.
- Answer all questions in the provided space directly on the test.
- **Failure to comply will result in loss of letter grade.**

This exam is 8 pages (without cover page) and 14 questions. Total of points is 140.

Grade Table (don't write on it)

Question	Points	Score
1	20	
2	15	
3	15	
4	5	
5	5	
6	5	
7	5	
8	20	
9	10	
10	15	
11	5	
12	5	
13	5	
14	10	
Total:	140	

1. On your answer sheet, write A-J and label each with abstraction or encapsulation.
  - (a) (2 points) A hides certain methods from users of the class by protecting them or making them private.
  - (b) (2 points) A hides whether an array or linked list is used.
  - (c) (2 points) E solves problem at implementation level.
  - (d) (2 points) E wraps code and data together.
  - (e) (2 points) A is focused mainly on what should be done.
  - (f) (2 points) E is focused on how it should be done.
  - (g) (2 points) A helps developers to design projects more easily.
  - (h) (2 points) A lets a developer use a class without worrying about how it's implemented.
  - (i) (2 points) A solves problem at design level.
  - (j) (2 points) E hides the irrelevant details found in the code.

2. (15 points) There are 3 major concepts when we think about OOP. What are they?

**Encapsulation**

---

**Inheritance**

---

**Polymorphism**

---

3. (15 points) There are 3 protection mechanisms we can use when we are building classes. What are they?

**Public**

---

**Private**

---

**Protected**

---

4. (5 points) Which of the protection mechanisms do we use so that a sub class (a class that inherits from another class) can gain access to sensitive portions of the class?

**Protected**

---

5. (5 points) What 3 protection mechanisms can we use if we use a **struct** instead of a **class**?

All the same ones! Structs differ from classes only in the fact that they default to public access rather than private.

**Public**

---

**Private**

---

**Protected**

---

6. (5 points) The snippet below shows an example of?

- A. Overloading
- B. Abstraction
- C. Polymorphism
- D. Encapsulation
- E. **A and C**

---

```
1  class MyNumsClass{
2      ...
3      int addnums(int a,int b){
4          return a + b;
5      }
6      int addnums(double a,double b){
7          return a + b;
8      }
9  };
```

---

7. (5 points) In the snippet below, what will line 20 (the last line) print to std out?

- A. 2 , 4
- B. 4 , 16
- C. **0 , 0**
- D. 'x' , 'y'

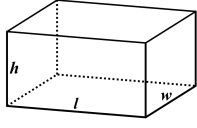
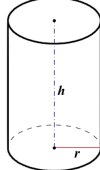
---

```
1  struct numClass{
2      int x;
3      int y;
4      numClass(){
5          x = y = 0;
6      }
7      int square_add(int x,int y){
8          x = x*x;
9          y = y*y;
10         return x + y;
11     }
12
13     void printXY(){
14         cout<<x<<" "<<y<<endl;
15     }
16 };
17 void main(){
18     numClass A;
19     cout<< A.square_add(2,4) << endl;
20     A.printXY();
21 }
```

---

8. (20 points) Prism Class Write a **Prism** class **definition** and **implementation** that will represent either a *cylindrical* prism or *rectangular* prism shape depending on how many parameters are used to instantiate the object. Assume all input values to be integers. Your prism class should calculate the volume correctly for each prism type. You should have a public *Volume* method that calls the appropriate private *Volume* method depending on the shape the object was instantiated as.

Example usage:

Rectangular Prism	Cylindrical Prism
	
$volume = l * w * h$	$volume = \pi r^2 * h$

```

1 Prism A(2,4);    // r=2,h=4
2 Prism B(2,4,6);  // w=2,h=4,l=6
3 cout<<A.Volume()<<endl; // prints: 100.53
4 cout<<B.Volume()<<endl; // prints: 48.0

```

### ANSWER:

```

1  #define PI 3.14159
2  class Prism{
3      int l;    // length
4      int w;    // width
5      int h;    // height
6      int r;    // radius
7
8      double VolumeCylinder(){
9          return PI * r*r * h;
10     }
11     double VolumeRectangle(){
12         return l * w * h;
13     }
14
15 public:
16     Prism(){l = w = h = r = 0;}
17     Prism(int l,int w, int h): l{l}, w{w}, h{h}{}
18     Prism(int h,int r): h{h}, r{r}{}
19
20     double Volume(){
21         if(r > 0){
22             return VolumeCylinder();
23         }else{
24             return VolumeRectangle();
25         }
26     }
27 };

```

9. (10 points) Given the following class definition, write a destructor for the class to delete the dynamically allocated memory.

---

```
1  class arrayClass{
2      int *A;
3      int size;
4      arrayClass(){
5          size = 10;
6          A = new int[size];
7      }
8      arrayClass(int s){
9          size = s;
10         A = new int[size];
11     }
12     ...
13     // destructor needed - define below outside of class
14 };
```

---

### ANSWER:

---

```
1  arrayClass::~~arrayClass(){
2      delete[] A;
3  }
```

---

10. (15 points) Write a class **definition** and **implementation** to represent a **Being**. Your class should include the following data members: age, height, weight, classification (alien or terrestrial) and color (gray,green,orange,purple,etc.). Include two constructors and methods to set and get each data member.

**ANSWER:**

---

```
1  class Being{
2      int age;
3      double height;
4      double weight;
5      string classification;
6      string color;
7  public:
8      Being(){
9          age = height = weight = 0;
10         classification = color = "";
11     }
12     Being(int a,double h, double w, string cls, string col): age{a}, height{h}, weight{w}, classification{
13         void setAge(int a){age = a;}
14         void setHeight(double h){height = h;}
15         void setWeight(double w){weight = w;}
16         void setClass(string c){classification = c;}
17         void setColor(string c){color = c;}
18
19         int getAge(){return age;}
20         double getHeight(h){return height;}
21         double getWeight(){return weight;}
22         void setClass(){return classification;}
23         void setColor(){return color;}
24
25     };
```

---

11. (5 points) Based on the question above, would you need to write a copy constructor for your **Being** class?
- A. Yes, we need to make sure we do a deep copy.
  - B. No, the deep copy will happen automatically.
  - C. **No, compiler handles this for us.**
  - D. Yes, deep copy or not we need a copy constructor always.
12. (5 points) Given the following:

---

```
1 Being& Being::operator=(const Being &other){
2     this.age = other.age;
3     height = other.height;
4     weight = other.weight;
5     this.classification = other.classification;
6     this.color = other.color;
7
8     return *this;
9 }
```

---

Which of the following are true?

- A. This method will error because you have to use the "this" operator for all assignments.
  - B. This method will error because you cannot return "this"
  - C. **This method works just fine.**
  - D. This method will not work because "other" should be "rhs"
13. (5 points) Which of the following applies to the **friend** operator.
- A. Gives specified class access to "private" data members of class where declared.
  - B. Gives specified class access to "protected" data members of class where declared.
  - C. Gives specified class access to "public" data members of class where declared.
  - D. **A and B only**
14. (10 points) Briefly describe the difference between a **class** and an **object**.

ANSWER:

Put simply:

A **class** is a "definition" of an *abstract data type* in which methods and data are packaged together. An **object** is an *instance* of a class that is memory resident and has a *state* (which can be changed through the methods provided in the class definition).

BONUS [15 points]: Given the following definition:

---

```
1 struct assignment{
2     string person_id;
3     string course;
4     float grade;
5 };
```

---

Assume there is a file called "grades.txt" with a 100 lines in it that look like:

```
M20273743 1063 88
M20172762 1063 89
M20597731 2143 48
M20573769 1063 98
...
M10573000 3498 50
```

Can you write the necessary code (not the entire program) to open the file and read it into an array of "assignments"? If so, show me below...

**For the answer, I'll provide a working program:**

---

```
1 #include <iostream>
2 #include <fstream>
3 using namespace std;
4
5 struct assignment{
6     string person_id;
7     string course;
8     float grade;
9 };
10
11 int main(){
12     assignment grades[100];
13     ifstream fin;
14     fin.open("grades.txt");
15
16     int i=0;
17     while(!fin.eof()){
18         fin>>grades[i].person_id;
19         fin>>grades[i].course;
20         fin>>grades[i].grade;
21         ++i;
22     }
23     return 0;
24 }
```

---