Midter Exam 1

Graded

Student

Total Points

74.75 / 100 pts

Question 1

Q1-EC 3 / 0 pts

✔ **+ 3 pts** Correct

Question 2

Q2 0 / 5 pts

✔ **– 5 pts** Incorrect

Question 3

Q3 5 / 5 pts

✔ **– 0 pts** Correct

Question 4

Q4 1.5 / 10 pts

Copy Constuctor

✔ **– 1 pt** Fails to correctly copy `theSize`

✔ **– 2 pts** Fails to correctly allocate new array

✔ **– 3 pts** Fails to correctly fill the array with copies of pointers to new `Product` objects (on the heap) (deep copy).

✔ **– 0.5 pts** Returns anything

Destructor

✔ **– 2 pts** Fails to iterate over dynamic array and delete `Product` objects

3 size of new array?

Question 5

Q5 5 / 5 pts

✔ **– 0 pts** Correct

Question 6

Q6 5 / 5 pts

✔ **– 0 pts** Correct

Question 7

Q7 **0** / 5 pts

✔ **− 5 pts** Incorrect

Question 8

Q8 **5** / 5 pts

✔ **− 0 pts** Correct

Question 9

Q9 **0** / 2 pts

✔ **− 2 pts** Incorrect

Question 10

Q10 **3** / 3 pts

✔ **− 0 pts** Correct

Question 11

Q11 **31.75** / 39 pts

operator<<

✔ **− 1 pt** Pointer in ranged for loop should be **pointer to const**

Constructor

✔ **− 0 pts** Correct

safe_to_add

✔ **− 1 pt** Pointer in loop should be **pointer to const**

add

✔ **− 4 pts** Incorrect **parameter passing,** e.g by-value or by constant-reference. Note copying address to pointer that is not "to const"

remove

✔ **− 0 pts** Correct

✔ **− 1 pt** index out of range

1

✔ **− 0.25 pts** Minor error

2 new is a keyword

Question 12

Q12 **15.5** / 16 pts

12.1 **A** **9.5** / 10 pts

✔ **– 0.5 pts** Poor use of emplace_back,. Making a copy.

12.2 **B** **6** / 6 pts

✔ **– 0 pts** Correct

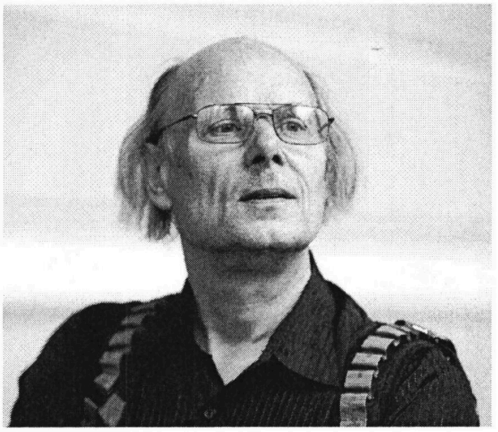
Name ___ ___ Net ID: _ _____

CS-UY 2124 - Object Oriented Programming
MID-TERM EXAM #1 - March 11th, 2025

- **Do not open this test booklet until you are instructed to do so**
- Duration: 1 hour, 15 minutes
- Do not separate any pages. Do not pull the test apart from the staple.
- Ensure your name and Net ID is printed at the top of every page.
- This is a closed book exam, no calculators, computers, or phones are allowed
- Anyone found cheating on this exam will receive a zero for the exam
- Anyone who is found writing after time has been called will receive a zero for this exam
- If you have a question please ask the proctor of the exam.
- Note that we have omitted any `#includes` or `using namespace std;` statements in all questions in order to save space and to save your time thinking about them. You may assume that all such statements that are needed are present. And you don't have to write them either!!!
- You also do not need to write any comments in any of your code.
- Please read all questions carefully! They may look familiar and yet be completely different.
- Answering the short-answer questions, in particular, requires that you read and understand the programs shown. You need to read them carefully if you are going to understand them.
- If a question asks you to write a class or a function and provides you with test code, **be sure your class / function works with that test code**. If the question provides you with sample output, then your answer should match that output.
- There is an **extra page at the end** in case you need it for any of the questions.
- Print your name and Net ID on the top of **EACH** page (yes, I know we said that already…).

Name ____ _____ Net ID: _ _____

1. EXTRA CREDIT: Who created C++? **Circle the correct answer** (including the item letter and the name).



A. Bill Gates

B. Sergey Brin

C. Guido van Rossum

D. Ada Lovelace

E. Alan Turing

F. James Gosling

G. Bjarne Stroustrup

H. Dennis Ritchie

I. Claude Shannon

J. None of the above

2. **(5 pts.)** Given the following code, what value would be output when the **`volume()`** method is invoked on the object **`container`**? Write the letter that corresponds to your answer in the box provided.

```
class Box {
public:
    int volume() const { return length * width * height; }

private:
    int length;
    int width;
    int height;
};

int main() {
    Box container;
    cout << "Container volume: " << container.volume() << endl;
}
```

A. 1	D. 3.14	Your answer:
B. 0	E. A compilation error occurs	B
C. Undefined result	F. none of the above	

Name _____ _____ Net ID: ___ _ _____

3. **(5 pts.)** Consider the **Box** class defined above. If a pointer to a **Box** named **b_ptr** exists in a program, which of the following expressions invokes the **volume()** method on the **Box** "pointed at" by **b_ptr**? Write the letter that corresponds to your answer in the box provided.

A. `b_ptr->volume()`	E. `*(b_ptr.volume())`	Your answer: A
B. `b_ptr<-volume()`	F. `(*b_ptr.volume())`	
C. `b_ptr>-volume()`	G. Any of the above	
D. `b_ptr.volume()`	H. None of the above	

Box x b_ptr

(*b_ptr).volume()

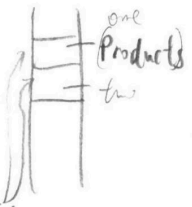
≡ b_ptr → volume()

Name _____ _____ Net ID: __ _____

4. **(10 pts.)** Given a class `Container` that has a dynamic array of pointers to objects of type `Product`, write a **destructor** and a **copy constructor** for the `Container` class. Note carefully the following:

- All the `Products` are on the heap.
- The `Container` object has the *only* pointers to the `Products`.
- The `Container` object has:
  - a field, `theSize`, storing the size of the dynamic array
  - a field, `products`, storing the address of the dynamic array.
- You don't know what is in a `Product`, but you do know that `Product` supports copy control.

heap




```
class Container{

private:
  int theSize;
  Product* products;
```

```
~Container(){
    delete [] products;
}

Container(const Container& rhs){
    products = new [] Product*;

    for (size_t i=0; i< theSize; ++i){
        products[i] =*(rhs.products+i);
    }
  return *this;
}
```

Name _____ _____ Net ID: _ ____

5. **(5 pts.)** Consider the existence of a class named `Pair`. Two `Pair` objects have been instantiated by the statements below:

```
Pair pt1(3.2, 4.6);
Pair pt2(9.1, -4.2);
```

A third `Pair` object is instantiated by the statement below.

```
Pair pt3 = pt1;
```

The statement above produces which of the following function calls? Write the letter that corresponds to your answer in the box provided.

A. `pt3.operator=(pt1)`	G. `the Pair copy constructor`	Your answer:
B. `pt1.operator=(pt3)`	H. `Pair(pt1.x, pt.y)`	G
C. `pt1.assignment=(pt3)`	I. `Pair(pt3.x, pt3.y)`	
D. `pt3.assignment=(pt1`	J. `None of the above`	
E. `Pair.operator=(pt1, pt3)`		
F. `Pair.assignment=(pt1, pt3)`		

Name _____ _____ Net ID: __ _____

6. **(5 pts.)** What is the result of building and running the program shown below? Write the letter that corresponds to your answer in the box provided.

```
class Outer {
public:
    class Inner {
    public:
        Inner(int val) : val(val) {} // Line A
    private:
        int val = 42;
    };

    Outer(int val) : mem(val) {}      // Line B
    int getInner() {                  // Line C
        return mem.val;               // Line D
    }

private:
    Inner mem;
};

int main() {
    Outer out(17);
    cout << out.getInner() << endl;
}
```

A. An address is displayed	F. Compilation error at line B	Your answer:
B. 17 is displayed	G. Compilation error at line C	H
C. 42 is displayed	H. Compilation error at line D	
D. Undefined behavior	I. Program compiles but crashes	
E. Compilation error at line A	J. None of the above	

Name _____ ___ Net ID: ___ ____

7. **(5 pts.)** What is the result of building and running the program shown below? Write the letter that corresponds to your answer in the box provided.

```
void func_b(double& b) {
    b += b;
}

double* func_a() {
    double b = 10.0;
    func_b(b);
    b++;
    return &b;
}

int main() {
    double* a = func_a();
    cout << *a << endl;
}
```

A. An address is displayed	E. Undefined behavior	Your answer: D
B. 10 is displayed	F. Compilation error	
C. 20 is displayed	G. Program compiles but crashes	
D. 21 is displayed	H. None of the above	

Name _____ Net ID: ___

8. **(5 pts.)** What is the result of building and running the program shown below? Write the letter that corresponds to your answer in the box provided.

```
void func_a (int& val) {
    const int* ptr = &val; // line A
    val = 27;              // line B
    cout << *ptr << ' ';   // line C
    *ptr = 54;             // line D
}

int main() {
    int num = 35;
    func_a(num);
    cout << num << endl;
    return 0;
}
```

A. Compilation error at line A	G. Output: 35 27	Your answer:
B. Compilation error at line B	H. Output: 27 54	
C. Compilation error at line C	I. Output: 35 27	D
D. Compilation error at line D	J. Output: 35 54	
E. Runtime error or undefined behavior at line D	K. None of the above	
F. Output: 35 35		

Name _____ Net ID: _ _ _____

9. **(2 pts.)** Consider the following C++ class definitions:

```
class Engine {
public:
    Engine(int hp) : horsepower(hp) {}
    int getHorsepower() const { return horsepower; }
private:
    int horsepower;
};

class Car {
public:
    Car(int hp) : engine(hp) {}
    int getCarHorsepower() const { return engine.getHorsepower(); }
private:
    Engine engine;
};

class Driver {
public:
    Driver(Engine* eng) : enginePtr(eng) {}
    int getEnginePower() const { return enginePtr->getHorsepower(); }
private:
    Engine* enginePtr;
};
```

Which of the classes demonstrates association? Write the letter that corresponds to your answer in the box provided.

A. The `Engine` class B. The `Car` class C. The `Driver` class D. More than one of the above E. None of the above	Your answer: D

Name _____ Net ID: _____

10. **(3 pts.)** Assume that the following struct and pointer are defined in your program:

```
struct Car {
    string brand;       // Car brand (e.g., Toyota, Ford)
    int year;           // Manufacturing year
    double engineSize;  // Engine size in liters
};

Car* my_car = new Car();
```

Write three lines of code that will set `my_car`'s `brand`, `year`, and `engineSize` values to "Honda", 2024, and 2.5 respectively.

```
my_car -> brand = "Honda";
my_car -> year = 2024;
my_car -> engineSize = 2.5;
```

Name _____ Net ID: _____

11. **(39 pts.)** Write a class named `Employee` that represents an employee in a company. Each `Employee` can have friends, which are other `Employee` objects. Each `Employee` object should contain

- a string **`employee_name`**,
- an integer **`employee_id`**, and
- a mechanism to track the employee's friends (known as a group).

Provide the definition of the `Employee` class. This is the ONLY class you need to write. In your class, you need to do the following:

- implement fields for the name and the id of the `Employee`,
- implement the `Employee` mechanism that tracks the friends, and
- overload the output operator for an `Employee` providing the information about the `Employee` (name, id, and the list of the names of all of the friends)

Implement the following methods

- add a method named **`add`** that adds a single `Employee` to an `Employee`'s friend group,
- add a method named **`safe_to_add`** that confirms:
  - that an `Employee` is not attempting to add itself to its own friend list (see rule #1 below), and
  - that an `Employee` does not previously exist in a group (see rule #2 below),

  This method should be called by the add method, described above, when the `Employee` is attempting to be added as a friend to any `Employee`. If either rule is violated, this method should fail and thus the attempt at adding should also fail.
- add a method named **`remove`** that removes a single `Employee` from it's friend's group (note that the order of the `Employees` in the group should be maintained). You should remove the `Employee` passed to this method.

Enforce the following rules

1. An `Employee` can not occur in its own group.
2. An `Employee` can not occur in the group more than once, (e.g. `EmployeeA` can occur in `EmployeeB`'s group only once, but could also occur in the group for `EmployeeC` and `EmployeeD`).
3. **Do not** use the Employee name and / or ID to determine if two Employee objects represent the same person. Yes, it does seem strange but somehow we have seen Employees with the same name and even the same ID! What a pain.
4. Groups are unrelated to each other. That is, if `EmployeeA` is added to the group for `EmployeeB`, it is not automatically added to any other group, nor is it reflexive (`EmployeeB` is not automatically added to `EmployeeA`'s group).
5. `Employee` additions or removals failures should not fail silently!

Note: This problem does not involve copy control or the heap.

This problem specification is continued on the next page...

Name _____ Net ID: __ _

Sample test code

You should consider the following code to test your implementation.

```
Employee e1("Juan de Pablo", 123456);
Employee e2("Martín Farach-Colton", 56734);
Employee e3("Daniel Katz-Braunschweig", 45456);
Employee e4("Peter DePasquale", 238213);
e1.add(e2);
e1.add(e2);  // Note that e2 can not occur twice in e1's group
e1.add(e3);
e1.add(e4);
e2.add(e1);
e2.add(e3);  // Note that e3 occurs for both e1 and e2
e3.add(e4);
e3.add(e1);
e4.add(e4);  // Note that e4 can not occur in its own group

e3.remove(e4);
e1.remove(e3);
cout << e1 << endl << e2 << endl << e3 << endl << e4;
```

Begin your answer here for the `Employee` class implementation.

```
class Employee{
    string employee_name;
    int employee_id;
    vector <Employee *> group;

    friend ostream& operator<<(ostream& os, const Employee& rhs){
        os << "Employee name: " << rhs.employee_name
        << ", id: " << rhs.employee_id << ". Groups: ";
        if (rhs.group.size() == 0){
            os << "none.";}
        else {
            for (Employee* each : rhs.group){
                os << each->employee_name << ", ";
            }
        }
        return os;} //op<<
```

Name ____ _____ Net ID: ___ _____

Continue your `Employee` class implementation here.

```
public:
    Employee (const string& name, int id): employee_name(name),
     employee_id (id) {}

    bool safe_to_add ( const Employee& new) const {
        if ( this == &new) return false;
        for ( Employee* each : group ){
            if (each ==&new){
                return false;
            }
        }
        return true;
    }
    bool add (const Employee & new) {
        if ( safe_to_add (new)) {
            group. push_back (&new);
            return true;
        }
        return false; }
    bool remove( const Employee & old) {
        for (size_t i=0; i<group.size(); ++i){
            if (group[i] ==&old){
                group[i] = nullptr;
                for (size_t j=i; j<group.size(); ++j){
                group [j]= group[j+1]
                }
```

25

Name _____ _____ Net ID: __ _____

Continue your `Employee` class implementation here.

```
            group.pop();
            return true;
          } // if
      } // for
      return false;
  } // remove

} ; // class
```

Name _____ _____ Net ID: __ _____

12. **(16 points total)** Write two functions:
- one will read a tab-separated (\t; tabs are whitespace) input data file that contains chemical element data and which populates a vector that stores the data structures that hold each row of data, and
- one that will display those vector items which match a filter requirement.

The Input File
Each line of the file is formatted as shown below. There is no header line in the input file. Each line of the input file should be represented as an instance of an `Element` struct (shown below). Note that there is no whitespace in the element / discoverer's names. (See the example data file.)

Input File Line Format (also see example below)
```
year<tab>element name-discoverer's_name<tab>weight
```

Example of the input file
```
1868   Helium-Jules_Janssen      4.00
2006   Tennessine-Joint_Institute_for_Nuclear_Research      294.00
1825   Aluminum-Hans_Christian_Ørsted   26.98
1669   Phosphorus-Hennig_Brand   30.97
1952   Einsteinium-Albert_Ghiorso252.00
1940   Plutonium-Glenn_T._Seaborg244.00
1735   Cobalt-Georg_Brandt 58.93
1807   Sodium-Humphry_Davy 22.99
1789   Uranium-Martin_Heinrich_Klaproth 238.03
```

Element Struct
```
struct Element {
  int year;
  string name;
  double weight;
};
```

Example of calling the `read_data` and `display_prior_to` functions from `main`
As an example, the functions could be called as:

```
int main() {
  vector<Element> my_data;
  ifstream ifs("input.txt");
  read_data(ifs, my_data);
  close(ifs);
  display_prior_to(my_data, 1850);
  ifs.close();
}
```

Example output
```
Aluminum-Hans_Christian_Ørsted - (1825) - 26.98
Phosphorus-Hennig_Brand - (1669) - 30.97
Cobalt-Georg_Brandt - (1735) - 58.93
Sodium-Humphry_Davy - (1807) - 22.99
Uranium-Martin_Heinrich_Klaproth - (1789) - 238.03
```

Name _____ Net ID: _____

`read_data` Function

Implement the `read_data` function. Your implementation should accept the input stream and a vector for the data. The function should populate the vector and return nothing.

```
void read_data(ifstream& ifs, vector<Element>& my_data){
    int year;
    string name;
    double weight;
    while (ifs >> year >> name >> weight){
        my_data.emplace_back({year, name, weight});
    }
}
```

Name _____ _____ Net ID: __ _____

`display_prior_to` Function

Implement the `display_prior_to` function. Your implementation should accept the data vector and an integer value. Print any vector items (output any reasonable format you wish, but include all values in the output of each item) which have a discovery year value that is less than the year parameter. The function should return nothing.

```
void display_prior_to(const vector<Element>& data, int cYear){
    for (size_t i=0; i< data.size(); ++i) {
        if (data[i].year < cYear){
        cout << data[i].name << " - C"
        << data[i].year << ") - "
        << data[i].weight << endl;
        }
    }
}
```

Name _____ Net ID: __ __ __

EXTRA SPACE IF NEEDED FOR ANY QUESTION