midterm1

● Graded

Student

Total Points

71 / 104 pts

Question 1

(no title) **4** / 4 pts

✔ **– 0 pts** Correct: Bjarne Stroustrup

Question 2

(no title) **5** / 5 pts

✔ **– 0 pts** Correct: `sq->displayArea()`

Question 3

(no title) **0** / 5 pts

✔ **– 5 pts** Line 3

Question 4

(no title) **0** / 5 pts

✔ **– 5 pts** Line 4

Question 5

(no title) **5** / 5 pts

✔ **– 0 pts** Correct: undefined

Question 6

(no title) **0** / 5 pts

✔ **– 5 pts** 2

Question 7

(no title) **5** / 5 pts

✔ **– 0 pts** Correct: 4, 3

Question 8

(no title) **5** / 5 pts

✔ **– 0 pts** Correct: `c1.operator+=(c2);`

Question 9

(no title) **5** / 5 pts

✔ **– 0 pts** Correct

Question 10

(no title) **8** / 8 pts

✔ **– 0 pts** Correct

Question 11

(no title) **9** / 12 pts

11.1 **A** **4** / 4 pts

✔ **– 0 pts** Missing parentheses for function definition

11.2 **B** **5** / 8 pts

✔ **– 2 pts** Failing to initialize non-primitive field in the initialization list

✔ **– 1 pt** Use of a **.size()** method

Question 12

(no title) **25** / 40 pts

Output Operator

✔ **– 0 pts** Correct

Constructor

✔ **– 0 pts** Correct

can_add

✔ **– 5 pts** Method should be **const**

✔ **– 2 pts** Check for adding self missing or incorrect

✔ **– 3 pts** Check for "not already your opponent" missing or incorrect

1 No op== defined

add_opponent

✔ **– 5 pts** Parameter passing

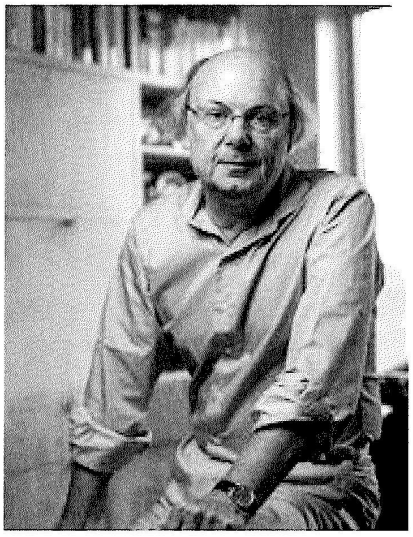
Name _____ _____ Net ID: _ _____

CS-UY 2124 - Object Oriented Programming
Exam One – October 22, 2024

- **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 are 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.
- Print your name and Net ID on the top of **EACH** page. (Yes, we already said that.)

Name _____ _____ Net ID: __ _____

1. **EXTRA CREDIT (3 points):** Who created C++? Fill in the circle that corresponds to your answer. Fill in only one circle.



- ○ Bill Gates
- ○ Sergey Brin
- ○ Guido van Rossum
- ○ Ada Lovelace
- ○ Alan Turing
- ○ James Gosling
- ● Bjarne Stroustrup
- ○ Dennis Ritchie
- ○ Claude Shannon
- ○ None of the above

2. **(5 pts.)** Consider the following class:

```
class Square {
public:
    Square(const string& name, double x) : name(name), x(x) {}
    void displayArea() const {
        cout << "Area of " << name << " is " << x * x;
    }
private:
      string name;
      double x;
};
```

(*sq).display

If a prototype for a function named print is declared as follows,

```
void print(const Square* sq);
```

If you are tasked with implementing the `print` function's definition, which expression would you use (in `print`'s function body) to invoke the `displayArea()` method within the `Square` class?

Completely fill the circle next to your Choice.

- ○ `sq.displayArea()`
- ● `sq->displayArea()`
- ○ `*(sq.displayArea())`
- ○ `sq<-displayArea()`
- ○ `*sq.displayArea()`
- ○ `*sq->displayArea()`
- ○ None of the above

Name _____ _____ Net ID: ____ __ _____

3. **(5 pts.)** Consider the following code. What would be the output when attempting to compile-and-run the program? Completely fill the circle next to your choice.

```
class Thing{
public:
    Thing(double x = 0): x(x) {}                    // line 1
    void display() {
        cout << x;                                  // line 2
    }
private:
    double x;
};
int main() {
    const Thing* th = new Thing;                    // line 3
    th->display();                                  // line 4
    delete th;                                      // line 5
}
```

- ○ 0
- ○ undefined
- ○ 15.2
- ○ Compilation error at line 1
- ○ Compilation error at line 2
- ● Compilation error at line 3
- ○ Compilation error at line 4
- ○ Compilation error at line 5

Name _____ _____ Net ID: _ _____

4. **(5 pts.)** Consider the following code. What would be the result when attempting to compile-and-run the program? Completely fill the circle next to your choice.

```
class Thing{
public:
    Thing(double x = 0): x(x) {}                    // line 1
    void display() {
        cout << x;                                  // line 2
    }
private:
    double x;
};
int main(){
    Thing th;                                       // line 3
    th = 2.718;                                     // line 4
    th.display();                                   // line 5
}
```

- ○ 0
- ○ undefined
- ○ 2.718
- ○ Compilation error at line 1
- ○ Compilation error at line 2
- ○ Compilation error at line 3
- ● Compilation error at line 4
- ○ Compilation error at line 5

Name _____ Net ID: _____

5. **(5 pts.)** Consider the code below. What would be the result when attempting to compile-and-run the program? Completely fill the circle next to your choice.

```
class Rectangle{
public:
    double getArea() const {
        return length * width;
    }
private:
    double length;
    double width;
};
int main() {
    Rectangle rec;
    cout << "Rectangle's area: " << rec.getArea() << endl;
}
```

- ○ 0
- ○ 1
- ○ 3
- ○ Fails to compile
- ● undefined
- ○ none of the above

6. **(5 pts.)** Consider the code below. What would be the output when attempting to compile-and-run the program? Completely fill the circle next to your choice.

```
int main() {
      int* numbers = new int[5];
      for (int i = 0; i < 5; i++) {
            numbers[i] = i;
      }
      numbers++;
      for (int i = 0; i < 2; i++) {
            numbers[i]++;
      }
      cout << *(numbers + 1) << " ";
}
```

- ○ 3
- ○ 1 2 3 4 5
- ○ 4
- ● 2
- ○ 0 2 3 3 4
- ○ 1

Name _____ _____ Net ID: __ _____

7. (5 pts.) Consider the code below. What would be the output when attempting to compile-and-run the program? Completely fill the circle next to your choice.

```
void compute(int& num) {
      cout << num++ * 2 << ", ";     // line 1
}
int main() {
      int x = 2;                     // line 2
      compute(x);                    // line 3
      cout << x << endl;
      return 0;
}
```

- ○ 3, 3
- ○ 3, 2
- ○ 2, 3
- ● 4, 3
- ○ Compilation error at line 1
- ○ Compilation error at line 2
- ○ Compilation error at line 3
- ○ Run-time error

Name _____ _____ Net ID: _____ _____

8. **(5 pts.)** Given the code below, what is the equivalent **function call** for the expression on the line marked **"THIS LINE"**, below? Completely fill the circle next to your choice.

```
class Complex{
public:
    Complex(double re=0, double im=0):re(re),im(im){}
    Complex& operator+=(const Complex& rhs) {
        re += rhs.re;
        im += rhs.im;
        return *this;
    }
    void display() {
        cout << "real: " << re << ", imaginary: " << im << endl;
    }
private:
    double re;
    double im;
};
int main() {
    Complex c1(2,3), c2(3,4);
    c1 += c2;  // THIS LINE
    c1.display();
}
```

O operator+=(c1, c2);

O c1 = operator+=(c1,c2);

● c1.operator+=(c2);

O c2.operator+=(c1);

O all of the above

O none of the above

9. **(5 pts.)** Using the same class `Complex` from above, and given a vector vc, defined as "`vector<Complex> vc;`", define a Complex number **`result`** and **using a ranged-for** (also known as the "for-each loop") use `result` to compute the sum of all the elements of the vector. Then display result. Note: don't define a function or re-declare the vector vc, just define `result`, implement the ranged-for loop and display the result.

```
Complex result;
for (const Complex& comp : vc){
    result += comp; }
result.display();
```

Name _____ ___ _____ Net ID: __ _____

10. **(8 pts.)** Consider the Student struct definition below:

```
struct Student {
    string firstname;
    string lastname;
    int year;
    double gpa;
};
```

Write a function parse_data that reads an input data file that contains entries of student information. The function will fill a vector of `Student`(s) that store the information, where each entry stores the information from one row from the input file.

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 a `Student`

```
firstname lastname year gpa
```

Example of the input file `students.txt`

```
John Smith 2 3.7
Michael Angelo 3 3.9
Jessica Lee 1 2.9
```

parse_data Function

Implement the **`parse_data`** function. Your implementation should accept the input stream and a vector for the data. The function must populate (i.e fill) the vector and return nothing.
You may assume the input stream has been correctly opened already!

Example of calling the `parse_data` function from `main`

As an example, the functions could be called as:

```
int main() {
    ifstream ifs("students.txt");
    if (!ifs) exit(1);

    vector<Student> vs;
    parse_data(ifs, vs);

    // Yeah and then we would print out the information...
}
```

Implement `parse_data` on the next page

Name _____ _____ Net ID: __ _____

Implement `parse_data` in the space below

```
void parse_data ( ifstream& ifs , vector<Student>& vs) {
    string fn ;
    string ln ;
    int year;
    double gpa;
    while ( ifs >> fn >> ln >> year >> gpa) {
        Student aStudent { fn, ln, year, gpa};
        vs. pushback ( aStudent) ; }
```

Name _____ _____ Net ID: ___ _____

11. **(12 pts.)** Given the **Student** struct from the previous question and a class **CourseSection** that has three fields (also known as data members) listed below.

```
string name;         // the name of the course
int numStudents;     // number of Students in the array
Student* students;   // pointer to a dynamic array of Student(s)
```

Write the following two functions.

a. **(4 pts.) Write a destructor for the CourseSection class**

```
~CourseSection {
    delete[] students; }
```

b. **(8 pts.) Write a copy constructor for the CourseSection class.**

```
CourseSection(const CourseSection& rhs) {
    name = rhs.name;
    numStudents = rhs.numStudents;
    students = new Student[rhs.students.size()]
    for (size_t index=0; index < rhs.students.size(); index++) {
        students[index] = rhs.students[index]; }
```

Name _____ _____ Net ID: _ ___ _____

12. **(40 pts.) For a chess tournament, you will <u>define a single class</u>**, `ChessMaster`, defined as follows

    - A `ChessMaster` has a name (`string`),
    - A `ChessMaster` has a ranking (`int`),
    - A `ChessMaster` has a city where he is located (`string`),
    - A `ChessMaster` has a country where he is located (`string`),
    - A `ChessMaster` has a collection of opponent `ChessMaster(s)` that it intends to play during a tournament. Note, this is an **association** between the chess master and his opponents.

    Provide the definition of the `ChessMaster` class. This is the ONLY class you need to write.

    Implement the following functions

    - an appropriate constructor for the `ChessMaster` class (see the test code below)
    - an output operator for a `ChessMaster` object providing the information about the `ChessMaster` object. This must include the `ChessMaster` name, ranking, city, country and the list of opponent `ChessMaster(s)` that it will face in the tournament, if applicable. (See example output below.)
    - a member function **can_add** which returns true if it is ok to add a `ChessMaster` to the collection of opponent `ChessMaster(s)`. This method will enforce the rules listed below.
    - a member function **add_opponent** which adds a `ChessMaster` to the collection of opponent `ChessMaster(s)`. Of course, this method uses the method can_add to determine if it is ok to do so.

    Enforce the following rules

    - A `ChessMaster` cannot be added to its own collection of opponents
    - An opponent `ChessMaster` may be added if, and only if:
        - Its ranking differs by less than "100" (i.e. 99 or smaller) from the ranking of the `ChessMaster` it's being added to, and
        - The two `ChessMaster(s)` are not from the same country.
    - This is a **reciprocal** relationship. That means that when `ChessMaster` A adds `ChessMaster` B to its collection of opponents, then `ChessMaster` B shall have `ChessMaster` A in its collection of opponents.
    - a `ChessMaster` cannot be added more than once to the collection of another `ChessMaster`,

- Note that a `ChessMaster` *may* be added to multiple `ChessMaster`(s), i.e. . `ChessMaster` A might be added to `ChessMaster` B, as well as `ChessMaster` C's collection of opponents.

- the **`add_opponent`** method should not fail silently. That means that if it fails, return `false`. If it succeeds, return `true`.

Note

- This problem does not involve copy control or the heap. **Do not** allocate a `ChessMaster` on the heap!
- **There is Sample test code and output on the next page, i.e. the back side of this page.**
- **My solution was about 30 lines.**

Name _____ _____ Net ID: _____

Sample test code

You should consider the following code to test your implementation:

```
ChessMaster p1("Gary Kasparov", 2851, "Baku", "Azerbaijan");
ChessMaster p2("Bobby Fischer", 2785, "Reykjavík", "Iceland");
ChessMaster p3("Emanuel Lasker", 2660, "Barlinek", "Poland");
ChessMaster p4("Paul Morphy", 2572, "New York", "USA");
ChessMaster p5("Magnus Carlsen", 2831, "Tonsberg", "Norway");
ChessMaster p6("Anatoly Karpov", 2617, "Ziatoust", "Russia");
ChessMaster p7("Jose Kapablanca", 2801, "Havana", "Cuba");

p1.add_opponent(p2);    // returns true
p1.add_opponent(p1);    // returns false, can't add to self
p2.add_opponent(p1);    // returns false, already added once
p2.add_opponent(p3);    // returns false, rating difference >100
p4.add_opponent(p6);    // returns true
p5.add_opponent(p6);    // returns false, rating diff >100
p7.add_opponent(p1);    // returns true

cout << p1 << endl;
cout << p2 << endl;
cout << p4 << endl;
```

Sample output

The following output was produced from executing our sample test code on our solution:

```
Gary Kasparov: 2851 (Baku, Azerbaijan) faces Bobby Fischer(Reykjavφk,
Iceland), Jose Kapablanca(Havana, Cuba),
Bobby Fischer: 2785 (Reykjavφk, Iceland) faces Gary Kasparov(Baku,
Azerbaijan),
Paul Morphy: 2572 (New York, USA) faces Anatoly Karpov(Ziatoust,
Russia),
```

Begin your answer to this question on the next page.

Name _____ _____ Net ID: ___ _____

```
class ChessMaster {
    friend ostream& operater<<(ostream& os, const ChessMaster & rhs);
    // defined outside the class, on the next page, no room to write here.
public:
    ChessMaster(const string& name, int ranking, const string& city, const string& country)
    : name(name), ranking(ranking), city(city), country(country) { }

    bool can_add(const ChessMaster & rhs){
        if(&rhs == *this) { return false; }
        if(rhs.ranking - ranking >= 100 || ranking - rhs.ranking >= 100) {
            return false; }
        if(rhs.country == country) { return false; }
        for(size_t index=0; index< op.size(); index++) {
            if(*op[index] == rhs) { return false; }}
        return true; }
    bool add_opponent(const ChessMaster & rhs){
        if(!can_add(rhs)) { return false; }
        op.push_back(&rhs);
        rhs.op.push_back(this);
        return true; }
```

Next Page →

```
private:
    string name;
    int ranking;
    string city;
    string country;
    vector<ChessMaster*> op;
}
```

```
ostream& operator<<(ostream& os, const ChessMaster& rhs){
    os<< rhs.name <<": " << rhs.ranking << "(" << rhs.city
     <<"," << rhs.country <<") faces";
    for(size_t index=0; index< rhs.op.size(); index++){
        os<< op[index]->name <<"(" << op[index]->city <<","
          << op[index]->country <<"),";}
    os<<endl;
    return os;}
```

Name _____ _____ Net ID: __ _____