CS-UY 2124 Exam 1

TOTAL POINTS

75 / 103

QUESTION 1

1 Question 1 **0 / 3**

✓ **- 3 pts** *Incorrect*

QUESTION 2

2 Question 2 **0 / 5**

✓ **- 5 pts** *Incorrect/Empty*

QUESTION 3

3 Question 3 **0 / 3**

✓ **- 5 pts** *Incorrect/Empty*

QUESTION 4

4 Question 4 **5 / 5**

✓ **- 0 pts** *Correct*

QUESTION 5

5 Question 5 **5 / 5**

✓ **- 0 pts** *Correct*

QUESTION 6

6 Question 6 **5 / 5**

✓ **- 0 pts** *Correct*

QUESTION 7

7 Question 7 **5 / 5**

✓ **- 0 pts** *Correct*

QUESTION 8

Question 8 10 pts

8.1 Question 8a **0 / 3**

✓ **- 2 pts** *Missing or incorrect pointer type*

✓ **- 2 pts** *Incorrect array type / initialization*

8.2 Question 8b **2 / 2**

✓ **- 0 pts** *Correct*

8.3 Question 8c **0 / 2**

✓ **- 2 pts** *Does not add to the int on the heap*

8.4 Question 8d **1 / 3**

✓ **- 2 pts** *Not correctly deleting array*

QUESTION 9

9 Question 9 **6 / 10**

✓ **- 2 pts** *Self assignment test missing or incorrect*

✓ **- 2 pts** *Incorrect return value*

QUESTION 10

10 Question 10 **18 / 18**

✓ **- 0 pts** *Correct*

parse_data

✓ **- 0 pts** *Correct*

count_entries

✓ **- 0 pts** *Correct*

QUESTION 11

11 Question 11 **28 / 34**

Output operator

✓ **- 0 pts** *Correct*

Constructor

✓ **- 0 pts** *Correct*

`is_paired_with`

✓ **- 4 pts** *Method must be const*

`pairs_with`

✓ **- 2 pts** *Inappropriate / incorrect additional tests*

1

Name Maksym Yemelyanenko Net ID: my2658

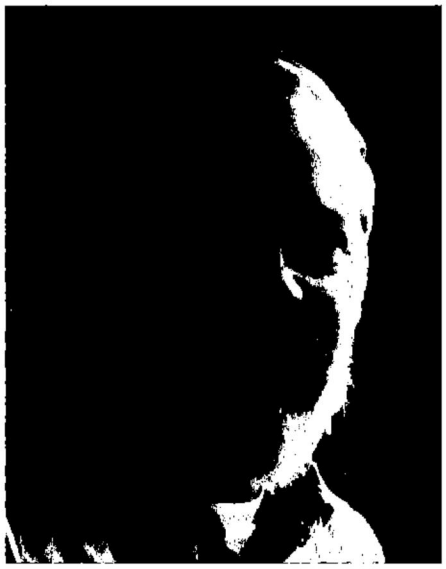
CS-UY 2124 - Object Oriented Programming
MID-TERM EXAM #1 - March 7th, 2023

- **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.
- Make sure 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.

Name Maksym Yemelianenko Net ID: my2658

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

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



2. **[5 pts]** Given a class called `Thing` and the code

```
Thing thingOne;
```

What function **call** is the following line equivalent to? Fill in only one circle.

```
Thing thingTwo = thingOne;
```

- ○ `Thing& Thing::operator=(const Thing& rhs)`
- ● `operator=(thingTwo, thingOne)  // (b)`
- ○ `thingOne.operator=(thingTwo)  // (c)`
- ○ Either (b) or (c), depending on how the programmer chose to implement the operator
- ○ None of the above because it is using the `Thing` copy constructor.
- ○ None of the above

Name Maksym Yemelianenko Net ID: my2658

3. **[3 pts]** The expression `*p.x` means the same thing as which option below? Fill in the circle that corresponds to your answer. Fill in only one circle.

- ● `p->x`
- ○ `*(p.x)`
- ○ all of the above
- ○ none of the above

4. **[5 pts]** Given a **dynamic array of integers** whose address is stored in the variable **data.**

The array holds the following ten values: 0, 1, 4, 9, 16, 25, 36, 49, 64 and 81.

Which of the following is equivalent to `&data[5]`?
(Note that there is ONLY ONE correct answer.)

- ○ 16
- ○ 25
- ○ 36
- ○ `*data+5`
- ○ `*(data+5)`
- ○ `&data+5`
- ◌ `&(data+5)`
- ● `data+5`
- ○ None of the above

5. **[5 pts]** Given a vector of strings, called `strVec`, use a ranged for (also known as the "foreach"), to print the items in the vector to the standard output, one per line.
(No, do not put this in a function.)

```
for (const string& s : strVec) {
    cout << s << endl;
}
```

Name Maksym Yemelianenko Net ID: my2658

✓ 6. **[5 pts]** .Given the following:

```
void foo(const int x) {
    int* const p = &x;      // line A
    x = 17;                 // line B
    cout << *p << ' ';      // line C
    *p = 28;                // line D
}

int main() {
    int y = 42;
    foo(y);
    cout << y << endl;
}
```

What is the result of compiling and running the above code? (Fill in only one answer)

- ○ The program will have a compilation error at line A
- ● The program will have a compilation error at line B
- ○ The program will have a compilation error at line C
- ○ The program will have a compilation error at line D
- ○ The program will have a runtime error (or undefined behavior) at line D.
- ○ The program will print out: 17 17
- ○ The program will print out: 17 42
- ○ The program will print out: 42 17
- ○ The program will print out: 42 42
- ○ The program will print out: 17 28
- ○ The program will print out: 42 28
- ○ None of the above.

Name Maksym Yemelianenko Net ID: my2658

7. **[5 pts]** Given the following code:

```
class Dragon {
public:
    Dragon(string val) : payload(val) {}
    void display() { cout << "Dragon payload: " << payload << endl; }

private:
    string payload;
};

class Falcon {
public:
    void display() {
        cout << "Falcon class ship\n";
        fly.display();
    }
private:
    Dragon fly;
};

int main() {
    Falcon heavy;
    heavy.display();
}
```

What is the result of compiling and running the above code?

- ○ Outputs:
  `Dragon payload: Roadster`
- ○ Outputs:
  `Falcon class ship`
- ○ Outputs:
  `Dragon payload: Roadster`
  `Falcon class ship`
- ○ Outputs:
  `Falcon class ship`
  `Dragon payload: Roadster`
- ● Compile time error
- ○ Run time error (or undefined behavior)
- ○ None of the above

Name Maksym Yemelianenko Net ID: My2658

8. **[10 pts]** Given:

a. Define a variable **daip** that points to a dynamic array of 100 int pointers.

```
int* daip = new int[100];
```

b. Fill the array with the addresses of 100 ints that you allocate on the heap. The ints will have values starting from 1 and going up to 100.

```
for (int i=1; i<=100; i++){
    daip[i-1] = new int(i);
}
```

c. Now modify those values by adding the index of the entry to the *value* that was stored on the heap, e.g. add 17 to the integer pointed to by `daip[17]`.

```
for (int i=0; i<100; ++i) {
    *(daip+i) += i;
}
```

d. Finally, free up all of the space that you allocated on the heap.

```
for(int i=0; i<100; ++i) {
    delete daip[i];
}
delete daip;
daip=nullptr;
```

Name Maksym Yemelianenko Net ID: my268

9. **[10 pts]** Given the following code:

```
class ShuttleCraft {
public:
    ShuttleCraft(string s) : s(s) {}
    // ... possibly other methods that you don't know about
private:
    string s;
    // ... possibly other fields that you don't know about
};

class Orville {
public:
    Orville(string commander, string s)
        : commander(commander), p(new ShuttleCraft(s)) {}
    ~Orville() { delete p; }
private:
    ShuttleCraft* p;
    string commander;
};
```

Implement an appropriate **assignment operator** for the class `Orville`. Write it below. Yes, it will make a "deep copy". And yes, the class `ShuttleCraft` supports copy control.

```
Orville& operator=(const Orville& rhs) {
    commander = rhs.commander;
    delete p;
    p = new ShuttleCraft(*rhs.p);
}
```

Name Maksym Yemelianenko Net ID: my2698

10. **[18 pts] Write two functions** as described below.
    - one will read a tab-separated (`\t`; tabs are whitespace) input data file that contains subway entry data and which fills a vector that stores the data structures that hold each row of data
    - one that will display how many items in the vector that 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 **`Entry`** struct. Note that there is no whitespace within the *borough*, *time*, or *contact* values. (See the example data file.)

```
borough\ttime\tcontact
```

You can assume an **`Entry`** struct is defined containing the 3 fields:
a `string`: `boro`, an `int`: `time`, and a `bool`: `con`.

parse_data Function

Implement the **`parse_data`** function. Your implementation should accept the input stream and a vector for the data. The function will fill the vector and return nothing. The `con` attribute will be set to `true` when the input row contains the value `c` for `contact` (indicating an entry with contact, i.e. using a metrocard) and `false` otherwise (indicating a non-contact entry).
You may assume the input stream has been correctly opened already!

count_entries Function

Implement the **`count_entries`** function. Your implementation should accept the data vector, a string, and a boolean. Print the number of vector items which have both a `boro` value that is equal to the `string` parameter and a `con` value that is equal to the `bool` parameter. Output any format you wish but include the borough and contact value. The function does not return a value.

Example of calling the **`parse_data`** and **`count_entries`** functions from `main`:

```
int main() {
    vector<Entry> entries;
    ifstream ifs("entries.txt");
    parse_data(ifs, entries);
    count_entries(entries, "MN", false);
    count_entries(entries, "BK", true);
}
```

Example input file

```
MN    1034  nc
BK    1602  c
MN    2209  c
QN    0842  nc
BX    0357  c
BK    0034  nc
MN    1322  nc
```

Example output

```
MN: 2 (non-contact)
BK: 1 (contact)
```

Name Maksym Yemelianenko Net ID: my2858

Implement the **parse_data** and **count_entries** functions below.

```
void parse_data(ifstream& ifs, vector<Entry>& ents) {
    string boro, con;
    Entry e;
    int time;
    while (ifs >> boro >> time >> con) {
        e.boro = boro;
        e.time = time;
        if (con == 'c') {
            e.con = true;
        } else {
            e.con = false;
        }
        ents.push_back(e);
    }
}

void count_entries(const vector<Entry>& ents, const string& boro,
                   bool con) {
    size_t amt = 0;
    for (const Entry& e: ents) {
        if (e.boro == boro && e.con == con) {
            amt += 1;
        }
    }
    cout << amt << ": " << boro << " " << (con ? "contact" :
                                           "non-contact") << endl;
}
```

Name Maksym Yemelianenko Net ID: my2658

11. **[34 pts] You will define a single class**, **`Player`**, defined as follows

- A Player has a number (`int`),
- A Player has a handedness value (`string`), and
- A Player has a collection of **`Players`** that are paired with this **`Player`**.

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

Implement the following functions

- an appropriate constructor for the **`Player`** class (see the test code below)
- an output operator for a **`Player`** object providing the information about the **`Player`** object. This must include the player's number, handedness and the numbers of the **`Player`**(s) that are paired with this **`Player`**, if applicable. (See example output below.)
- a method **`is_paired_with`** that returns `true` if two Players are already paired together, and `false` otherwise.
- a method **`pairs_with`** which pairs the **`Player with another Player`**.

Enforce the following rules

- a **`Player`** can only pair with another **`Player`** when they have opposite handedness.
- Pairing is a **reciprocal** relationship. That means that when Player A pairs with Player B, then Player B becomes paired with Player A.
- a **`Player`** ~~cannot be added more than once to the "pairs" collection for another~~ **`Player`**, but *can* pair with more than one **`Player`**. So Player A might pair with both Players B and C, but can only pair *once* with, for example, Player B!.
- the **`pairs_with`** method should not fail silently. That means that if the pairing fails, return `false`. If it succeeds, return `true`.

Note

- This problem does not involve copy control or the heap. **Do not** allocate a Player on the heap!
- **`Players`** get to pick their own numbers! So, two *different* Players could have the *same* number! And they could even pair with each other! Yes, very confusing. Do not compare two **`Player`** objects by their number and handedness to determine if they are the same object!

There is Sample test code and output on the next page

Name Maksym Yemelyanenko Net ID: my26s

Sample test code

You should consider the following code to test your implementation:

```
Player p1(8, "right");
Player p2(72, "left");
Player p3(22, "right");
Player p4(15, "right");
Player p5(30, "left");
Player p6(1, "right");
Player p7(8, "left");    // Yes, a second player with the number 8!


p1.pairs_with(p2);       // returns true
p1.pairs_with(p1);       // returns false
p2.pairs_with(p1);       // returns false
p4.pairs_with(p2);        // returns true
p4.pairs_with(p1);       // returns false
p5.pairs_with(p6);       // returns true
p7.pairs_with(p1);       // returns true


cout << p1 << endl;
cout << p2 << endl;
cout << p3 << endl;
cout << p4 << endl;
cout << p5 << endl;
cout << p6 << endl;
cout << p7 << endl;
```

Sample output

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

```
Player 8 (right) pairs with 72 8
Player 72 (left) pairs with 8 15
Player 22 (right)
Player 15 (right) pairs with 72
Player 30 (left) pairs with 1
Player 1 (right) pairs with 30
Player: 8 (left) pairs with: 8
```

Begin your answer to this question on the next page.

Name Maksym Yemelianenko Net ID: my268

```
class Player {
    friend ostream& operator<<(ostream& ost, const Player& p) {
        ost << "Player " << p.num << " (" << p.hand << ");
        if (p.pairs.size() > 0) {
            ost << " pairs with";
            for (const Player *pl : p.pairs) {
                ost << " " << pl->num;
            }
        }
        return ost;
    }
private:
    int num;
    string hand;
    vector<Player*> pairs;
public:
    Player(int num, const string& hand) : num(num), hand(hand) {};
    bool is_paired_with(const Player& oth) {
        for (const Player* p : pairs) {
            if (p == &oth) {
                return true;
            }
        }
        return false;
    }
```

Name Maksym Yemelianenko Net ID: my265

```
bool pairs_with(Player& p) {
    if (this == &p) {
        return false;
    }
    if (hand == p.hand) {
        return false;
    }
    if (is_paired_with(p)) { return false; }

    pairs.push_back(&p);
    p.pairs.push_back(this);
    return true;
}
}
```

1

Name Maksym Yemelianenko Net ID: my2658