Midterm

● Graded

Student

Rentian Dong

Total Points

40 / 55 pts

Question 1

Question 1 - class definition

13 / 15 pts

- ✔ **+ 1 pt** Correct class prefix and use of public and private
- ✔ **+ 3 pts** Good choice of private member variables with syntactically correct declarations
- ✔ **+ 2 pts** Function signature and definition for constructor
- ✔ **+ 2 pts** Function signature and definition for recharge
- ✔ **+ 2 pts** Function signature and definition for togglePower
- ✔ **+ 4 pts** Function signature and definition for usePhone
- ✔ **− 1 pt** usePhone doesn't check if phone is on
- ✔ **− 1 pt** usePhone doesn't update battery life
- **− 1 pt** usePhone doesn't turn phone off when needed
- ✔ **+ 1 pt** Rationale for no destructor
- **+ 1 pt** XC: Correct declaration for overloaded insertion
- **+ 1 pt** XC: Good implementation of overloaded insertion

Question 2

Question 2 - declarations

2 / 6 pts

+ 3 pts Declaration for *

✔ + 3 pts Declaration for +

− 1 pt Declaration is in the wrong place

− 1 pt Incorrect syntax for function names

− 0.5 pts Missing/incorrect use of const

✔ − 1 pt Incorrect parameter declaration

1 need to specify parameter type (const Point3D&)

− 1 pt Incorrect return type

− 1 pt Incorrect function name

− 0.5 pts * should return a double instead of an int so no precision is lost

+ 0 pts No answer / no correct code

Question 3

Question 2 - definitions and main

6 / 9 pts

- ✔ **+ 2 pts** * definition
- ✔ **+ 2 pts** + definition
- ✔ **+ 1 pt** Declaration and initialization of p1 and p2
- ✔ **+ 2 pts** Declaration and initialization of p1*p2
- ✔ **+ 2 pts** Declaration and initialization of p1+p2
  - **− 1 pt** Incorrect scope resolution
  - **− 0.5 pts** Incorrect syntax for constructor call
  - **− 0.5 pts** friend shouldn't be repeated in the definition
  - **+ 0 pts** No answer / no correct code
- 💬 **− 3 pts** Point adjustment

2. -.5 need to specify return type and parameter type
3. -.5 need to declare types for p1 and p2
4. -.5 these should be doubles so you don't lose precision
5. -1 need to declare type for p4
6. -.5 these should be doubles so you don't lose precision

Question 4

Question 3 - definitions

6.5 / 9 pts

+ 0 pts No correct code

✔ + 5 pts Insert

✔ + 4 pts Resize

– 1 pt Insert doesn't allocate a new memory under the correct conditions

✔ – 0.5 pts Insert doesn't (correctly) update cap_ and/or size_

8 new array should have just one additional spot for new value

– 1 pt Insert doesn't correctly allocate new memory

– 1 pt Insert doesn't correctly deep copy contents when new memory is allocated

– 1 pt Insert doesn't free old memory and/or update data_

✔ – 1 pt Insert doesn't put val in correct spot in every case

7 also need to insert the val if cap_>size_

✔ – 1 pt Resize doesn't check for the correct relationship between size_ and cap

9 compare cap to size_ instead of cap_

– 1 pt Resize doesn't update cap_

– 1 pt Resize doesn't correctly create a new array and copy old contents into it

– 1 pt Resize doesn't delete old array and update data_

– 1 pt Resize *does* update size_

– 1 pt Missing or incorrect scope resolution

Question 5

Question 3 - declarations

6 / 6 pts

+ 0 pts No correct syntax or descriptions

✔ **+ 1 pt** Destructor syntax

✔ **+ 1 pt** Destructor description

✔ **+ 1 pt** Copy constructor syntax

✔ **+ 1 pt** Copy constructor description

✔ **+ 1 pt** Assignment operator syntax

✔ **+ 1 pt** Assignment operator description

– 0.5 pts Only mentions resetting member variables for destructor, not cleaning up heap

– 1 pt Doesn't describe how the redefined copy constructor and assignment operator differ from the default ones

Question 6

Question 4 - output

6.5 / 9 pts

✔ **+ 3 pts** Part 1: 13 15 13 13

✔ **+ 3 pts** Part 2: 2 2 3 3 5 5 10 10

+ 3 pts Part 3: 1 16 6 4

+ 0 pts No correct output

+ 0.5 pts .5 Part 3 partial credit

Question 7

Question 4 - code modification

0 / 1 pt

+ 1 pt delete [] arr or delete [] p at end of part2

– 0.5 pts Incorrect code modification that causes a problem

✔ **+ 0 pts** No answer / incorrect answer

Midterm

Friday, October 31, 2025

CSCI 60
Krehbiel

Overview. This midterm tests your understanding of OOP and memory management in C++. There are 4 questions; you have 65 minutes for the exam. You may use any blank page for scratch work; if you write answers other than the corresponding question page, indicate which page number contains your answer. You may assume all starter code and code you write uses the standard namespace and includes the iostream library; you don't need to add these or other libraries.

Honor pledge. Print and sign your name below to affirm that you agree to complete this exam entirely on your own and without assisting others, and that you will not communicate about the exam until after exam day.

Printed Name: Rentian Dang (Mike)

Signature: Rentian Dang (Mike)

[Scratch work / extra space for answers]

[Scratch work / extra space for answers]

1 Phone (15 points)

On the next page, define a class that represents a phone. Your class should keep track of the name of the phone's owner, whether it is on, and its remaining battery life. It should include the following member functions, which may be defined inline:

1. Write a 1-argument constructor that specifies the owner according to the parameter and initializes the phone to be fully charged and off.
2. Write a 1-argument function `usePhone` that does nothing if the phone is off. If it's on, it depletes the battery by an amount specified by the parameter; if this amount is greater than the battery life left, the battery life goes to zero and the phone turns off.
3. Write a function `recharge` that returns the phone (on or off) to full battery life.
4. Write a function `togglePower` that turns the phone off it is on and on if it is off.
5. Write a one-sentence comment about of why your class should or should not have a destructor.

Extra credit: Overload the insertion operator for your class; only declare it for partial extra credit.

[Your answer to Question 1]

```
class Phone{
 Private:
string name_;
bool status_; //true when on  false when off
 int battery_; // from 0 to 100
 Public:
     Phone(string name):name_(name), status_(false), battery_(100){}
     Void usePhone(int use){
        Int Final = battery_ - use;
        if (Final<0||Final=0){
            battery_ = 0;
            status_ = false;
        }
     }
     void recharge(){
          battery_ = 100;

     }
     void togglePower(){
        status_ = (!status_);
     }
};
```

// It should not have destructor, because there is nomemory on the heap

2 3D Point (15 points)

The following skeletal class represents a point in 3-dimensional space. Declare two operators below (in `point3d.h`) and define them on the following page (in `point3d.cpp`) as follows.

1. Overload * as a member function to calculate the dot product of two 3D points. The dot product of 3D points (a_1, a_2, a_3) and (b_1, b_2, b_3) is the quantity $a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3$.

2. Overload + as a non-member function to calculate the sum of two vectors. You may declare your function as a friend or not, and you may define accessors if you need them. The sum of 3D points (a_1, a_2, a_3) and (b_1, b_2, b_3) is the point $(a_1 + b_1, a_2 + b_2, a_3 + b_3)$.

Then write a simple program on the following page (in `main.cpp`) that does the following:

3. Declare a `Point3D` object `p1` representing the point (1,2,3), and declare another `Point3D` object `p2` representing the point (-1,2,4).

4. Declare a variable and use your overloaded operator to initialize it to be the dot product of `p1` and `p2`, and then declare another variable and use your overloaded operator to initialize it to be the sum of `p1` and `p2`.

[point3d.h]

```
class Point3D {
public:
  Point3D(double x, double y, double z) : x(x), y(y), z(z) {}
  Point3D() : Point3D(0,0,0) {}
```

friend Point3D operator+(const &lhs, const &rhs){}

```
private:
  double x;
  double y;
  double z;
};
```

[point3d.cpp]

```
#include "point3d.h"
Point3D :: operator*(const &rhs){const
    return (x * rhs.x + y*rhs.y+ z*rhs.z);
}
Point3D operator+(const &lhs, const &rhs){
      int X = lhs.x + rhs.x;
      int Y = lhs.y + rhs.y;
      int Z = lhs.z + rhs.z;
      Point3D result(X, Y, Z);
      return result;
}
```

[main.cpp]

```
#include "point3d.h"

int main() {
    p1(1,2,3);
    p2(-1,2,4);
 int p3 = p1 * p2;
    p4 = p1 + p2;
}
```

3 Resizable Array (15 points)

Below is a modified partial implementation of our `GArray` class called `RArr`. The primary modification is that there is a new member variable `cap_` representing the size of the dynamically allocated array, and the new class invariant allows capacity to be larger than the number of entries in the array. A new constructor allows for a client to declare an empty `RArr` object with a pre-allocated array. Continue developing this class by defining two functions on the following page (both of which must maintain the invariant and not orphan any memory) and then declaring three additional functions in the class below.

1. Define the `insert` function declared below. The logic should be similar to our implementation of `insert` for `GArray` in that it moves data from a full dynamic array to a new dynamic array with one additional spot for the new value, but it should be updated to reflect that now capacity may be larger than size, and in that case no new dynamic array is required.

2. Define the `resize` function declared below. If the new capacity is smaller than the current size, the function should do nothing. Otherwise, it should copy all the current contents into a new dynamically allocated array with the specified capacity.

3. Declare but do not define a destructor, copy constructor, and assignment operator. Write a one-sentence comment for each of these describing why it is necessary and it should do.

[rarr.h]

```
class RArr {
public:
  RArr(size_t cap) : size_(0), cap_(cap), data_(cap==0?nullptr:new int[cap]) {}
  void insert(int val);
  void resize(size_t cap);
  ~RArr();
  RArr(const RArr& other);
  RArr &operator=(const RArr &rhs);
// Destructor cleans up all the memory on the heap to prevent orphaned memory
// Copy constructor allows deep copy from one to another, which prevent shallow
  copy and give the copy/value a new different address
// assignment operator allows us to redefine an operator, it is necessary
      because we need to use the operator to return value or
private:                                    modify our RArr variables.
  size_t size_;
  size_t cap_;
  int *data_;
  // INVARIANTS: cap_>=size_, and data_ addresses a dynamic array of cap_ ints
  //             with size_ ints stored in data_[0],...,data_[size_-1]
};
```

it is necessary because it

[rarr.cpp]

```
void RArr::insert(int val){
    if(cap_ == size_){
        cap_ *= 2;
    }
    int* tempArray = new int[cap_];
    for (int i=0; i< size_; i++){
        tempArray[i] = data_[i];
    }
    tempArray[size_++] = val;
    delete[] data_;
    data_ = tempArray;
}
void RArr::resize(size_t cap){
    if(cap_ < cap){
        int* tempArray= new int[cap];
        for(int i=0; i<size_; i++){
            tempArray[i]= data_[i];
        }
        delete[] data_;
        data_ = tempArray;
    }
}
```

4 Tracing (10 points)

This question has you trace memory to determine output for the code on the following page. Additioanlly, add any code needed to avoid orphaning memory. Write the output below and add your code directly to the provided code in the appropriate place.

Part 1: 13 15 13 13

Part 2: 2 2 3 3 5 5 10 10

Part 3: 3 8 3 4

```
void part1() {
  int x = 5;
  int y = 10;
  int *p = &x;
  int *q = &y;
  x = y+1;
  y = *p+4;
  q = &x;
  *q = x+2;
  p = q;
  cout << "Part 1: " << x << " " << y << " " << *p << " " << *q << endl;
}

void part2() {
  int n = 4;
  int *arr = new int[n];
  for (int i=0; i<n; i++) {
    arr[i] = i+2;
  }
  int *p = arr;
  *(p+2) = *p + *(p+1);
  *(arr+3) = *(p+2)*2;
  cout << "Part 2: ";
  for (int i=0; i<n; i++) {
    cout << arr[i] << " " << p[i] << " ";
  }
  cout << endl;
}

int* fn(int a, int *b, int& c) {
  a += 2;
  *b += 2;
  c += 2;
  return b;
}

void part3() {
  int nums[4] = {1,2,3,4};
  int bonus = 10;
  int *p = nums;
  int *result = fn(p[0],p+1,bonus);
  *result += bonus;
  nums[2] = 6;
  cout << "Part 3: ";
  for (int i=0; i<4; i++) {
    cout << *(p+i) << " ";
  }
  cout << endl;
}

int main() {
  part1();
  part2();
  part3();
  return 0;
}
```

[Scratch work / extra space for answers]