Final Exam S24 Final V2

CSCI 13500: Software Analysis and Design 1

Hunter College, City University of New York

May 22, 2024, 11:30 AM - 1:30 PM, North Building Auditorium

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of a provided cheat sheet.
- When taking the exam, you may bring pens and pencils.
- Scratch paper is provided. For your convenience, you may take the scratch paper and cheat sheet off. But make sure not to put solutions to the scratch paper.
- You may not use a computer, calculator, tablet, phone, earbuds, or other electronic device.
- **Do not open this exam until instructed to do so.**
- If you earn a D in the class and would rather have an F, put an X in this box. This will not affect your grade if you earn a C or better. If you have already elected to take a P/NC you probably don't want to do this.

  ☐

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

I understand that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.
Name:
EmpID:
Email:
Signature:

1 (30 points) Answer the following questions.

(1) Given `string animals[] = {"hare", "tortoise", "elephant"}`, what is animals[2].substr(5, 3)?

(2) Given Fish class, declare that class Shark as a subclass of Fish class with public inheritance.

(3) Write statement to generate a random integer in [3, 11].

(4) Suppose data member `patterns` of a Hare object is {1, -1, 2}. The following is a definition of `move` method.

```
void Hare::move() {
    int index = rand() % patterns.size();
    int stepsToMove = patterns[index];
    position += stepsToMove;
}
```

Suppose rand() generates a random integer 10, and the value of data member `position` of an object is 2. After calling `move` method, what is the value of `position`?

(5) Write a unix command to compile `Hare.cpp` which has no main function to generate `Hare.o`.

(6) What is the value of `1 + 5 / 2 % 3` in C++?

(7) Write **header** of a function called concat, given an array of chars with *size* many elements, return a string concatenating all the characters in the given array.

(8) Given `int grades[] = {86, 77, 96, 81, 25};` What is the value of `*(grades + 1)`?

(9) Declare a double-type pointer p. Apply a dynamically allocated memory to consecutively hold 20 double numbers, and put its initial address to p.

(10) Suppose we have main function defined as follows.

```
int main() {
   int n = 2;
   foo("hello", &n);
   return 0;
}
```

What is the **header** of function foo? Suppose its return type is void.

(11) What is output for the following code?

```
    int a = 6;
    int* p = &a;
    a += 2;
    cout << *p << endl;
```

(12) What the output of the following code?

```
void foo(int size) {
    for (int numAsts = 1; numAsts <= size; numAsts += 2) {
        for (int i = 0; i < (size - numAsts)/2; i++)
            cout << " ";

        for (int i = 0; i < numAsts; i++)
            cout << "*";

        cout << endl;
    }
}
```

(13) What is the output of the following code?

```cpp
#include <iostream>
#include <string>
using namespace std;

int foo(string input, char ch, char ch2);

int main() {
    cout << foo("abc a", 'a', 'b') << endl;
    return 0;
}

int foo(string input, char ch, char ch2) {
    int num = 0;
    for (int i = 0; i < input.size(); i++) {
        if (input[i] != ch && input[i] != ch2)
            num++;
    }

    return num;
}
```

(14) What is the output for the following code?

```cpp
vector<int> nums;

for (int i = 1; i < 6; i++)
    nums.push_back(i);

int sum = 0;
for (int i = 0; i < nums.size(); i++)
    if (nums[i] % 2 == 0)
        sum += nums[i];

cout << sum << endl;
```

(15) What is the output of the following code? Assume that all necessary libaries are included and namespace is properly used.

```
void foo(vector<int>& v, int index, int value);

int main() {
    vector<int> v = {2, 3, 1};
    foo(v, 2, 7);

    for (int i = 0; i < v.size(); i++)
        cout << v[i] << " ";
    cout << endl;
    return 0;
}

void foo(vector<int>& v, int index, int value) {
     if (index >= 0 && index < v.size())
        v[index] = value;
}
```

2 (15 points) Answer the following questions.

1. Define function `percentage`, for an given array of characters with its size, return the percentage of characters that are either 'A' or 'B' in this array.

   For example, call the function with array with values 'A', 'C', 'B','D', 'F', 'C'. The size of array is 6, and there are two occurrences of 'A' or 'B'. The return is 33.3333.

   Warning: C++ is case-sensitive programming language.

2. **Provide** definition of class Date, which contains public integer members

```
year
month
```

The value in `month` should NEVER be > 12 or < 1, where 1 represents January and 12 represents December.

(a) **Define** a NON-member function `add_month` which, given Date object `curr` and number of integer representing `num_months`, return Date object representing the date after moving forward `num_months` from `curr`. For simplicity, **assume** that `num_months` is non-negative.

(b) Examples:

i. Suppose `curr` has `year` 2021 and `month` 12, which represents December 2021.

ii. Suppose `num_months` is 25.

iii. Call `add_month` function on `curr` and `num_months`, return `Date` object with data members `year` 2024 and `month` 1, which represents January 2024.

(c) Hints: one year has 12 months. Then 25 months equals 2 year and 1 month. What if the sum of the month of current date, say 12, with additional months, say 1, is larger than 12?

3 (10 points) Programming exercise on pointer

1. A Coord2D object represents a point in a 2-dimensional plane, where data members `x` and y are the x- and y-coordinate of that point, respectively. Do not mix point in geometry with pointer in C++.

```
class Coord2D {
public:
    double x;
    double y;
};
```

The slope of a line connecting two points is defined as $\frac{\texttt{y of the second point - y of the first point}}{\texttt{x of the second point - x of the first point}}$. If two points have the same value for x-coordinates, then the slope is defined as infinity, denoted by `std::numeric_limits<double>::infinity()` in C++.

For example, given Coord2D object a whose x is 1 and y is 2, Coord2D object b whose x is 6 and y is 3, the slope of the line connecting a and b is $\frac{3-2}{6-1} = 0.2$.

Given Coord2D object c whose x is 1 and y is 3, the slope of the line connecting a and c is infinity.

Define function `slope`, given two **pointers** to Coord2D objects, return the slope of the line connecting the two pointed Coord2D objects.

2. Write the following statements in main function. No need to include libraries or other parts of main function.

   - Define a as a Coord2D object with x-coordinate 1 and y-coordinate 2.
   - Define b as a Coord2D object with x-coordinate 6 and y-coordinate 3.
   - Find out and print the slope of the line connecting a and b.

4 (10 points) Write codes of vector

Define a function called `choose`, for a vector `v` of integers and `left` and `right` as integers, return a vector with all the elements from `v` that are in the range of [left, right], in the same order.

For example, given a vector of integers with elements 12, 3, 6, 7, 5 and left 3 and right 6, the return is a vector with elements 3, 6, 5.

5 (15 points) Define class for pentagon shape.

1. Each regular pentagon has 5 same-length sides.

   Assume that Pentagon.hpp is provided where data member **side** is defined as double type. **Your job** is to define the following constructors and methods in Pentagon.cpp. Suppose libraries are properly included in Pentagon.cpp.

2. Define the default constructor, initialize data member **side** to be 1.

3. Define a non-default constructor, which takes formal parameters side, a double type.

   (a) If given parameter side is positive, use it to initialize data member **side**, otherwise, initialize data member **side** by 1.

4. Define method **setSide**, if given parameter side is positive, use it to set data member **side**.

5. Define method **getPerimeter**, which returns 5 times **side**, the sum of all sides.

6. Define method **getArea**, which returns the area, calculated by $\frac{1}{4}\sqrt{5(5+2\sqrt{5})}side^2$, square root can be calculate by `sqrt` function from `cmath` library.

   double sqrt (double x);

Define **PentagonTest.cpp**, do the following:

1. Create a Pentagon object named **pente** from its default constructor.

2. Print out the area of **pente**.

3. Reset the side of **pente** to be 2.

6 (10 point) Define a subclass.

Here are part of Person.hpp of Person class.

```
class Person {
public:
    Person(string name, int age); //non-default constructor of Person class
    ...//omit other constructors and methods
private:
    string name;
    int age;
};
```

1. Declare Student as a subclass of Person. Each student is a person, with additional data member **majors**, a vector of strings, to describe majors.

2. Define non-default constructor of Student, given name (a string), age (an integer), and major (a string), instantiate a student as a person with name and age, and add major to data member **majors**.

   (a) You can invoke constructors from super class.

   (b) If major is not an empty string, use `push_back` method to add major to data member **majors**, otherwise, add "undecided" to data member **majors**.

3. Define method **getNumMajors** to return the number of majors of a student.

7 (10 points) Define recursive function

Define a recursive function, for an given array of integers, return the maximum integer. Note that the size of an array in C++ cannot be zero.

For example, suppose the array of strings has elements 2, 3, 1, the return is 3.

Hint: what if the array has only one element? When the array has more than one element, how to find out the maximum element in a subarray?

Warning: If you do not use recursion, you will not get any point. No repetition statement is allowed in this function.

Variable and Constant Definitions

Type *Name* *Initial value*

```
int cans_per_pack = 6;

const double CAN_VOLUME = 0.335;
```

Mathematical Operations

```
#include <cmath>
```

`pow(x, y)`	Raising to a power	x^y
`sqrt(x)`	Square root	$\sqrt{x}$
`log10(x)`	Decimal log	$\log_{10}(x)$
`abs(x)`	Absolute value	$\lvert x \rvert$
`sin(x)` `cos(x)` `tan(x)`	Sine, cosine, tangent of x (x in radians)	

Selected Operators and Their Precedence

(See Appendix B for the complete list.)

`[]`	Array element access
`++ -- !`	Increment, decrement, Boolean *not*
`* / %`	Multiplication, division, remainder
`+ -`	Addition, subtraction
`< <= > >=`	Comparisons
`== !=`	Equal, not equal
`&&`	Boolean *and*
`||`	Boolean *or*
`=`	Assignment

Loop Statements

Condition

```
while (balance < TARGET)
{
   year++;
   balance = balance * (1 + rate / 100);
}
```

Executed while condition is true

Initialization *Condition* *Update*

```
for (int i = 0; i < 10; i++)
{
   cout << i << endl;
}
```

Loop body executed at least once

```
do
{
   cout << "Enter a positive integer: ";
   cin >> input;
}
while (input <= 0);
```

Conditional Statement

Condition

```
if (floor >= 13)
{
   actual_floor = floor - 1;
}
else if (floor >= 0)
{
   actual_floor = floor;
}
else
{
   cout << "Floor negative" << endl;
}
```

Executed when condition is true

Second condition (optional)

Executed when all conditions are false (optional)

String Operations

```
#include <string>
string s = "Hello";
int n = s.length(); // 5
string t = s.substr(1, 3); // "ell"
string c = s.substr(2, 1); // "l"
char ch = s[2]; // 'l'
for (int i = 0; i < s.length(); i++)
{
   string c = s.substr(i, 1);
   or char ch = s[i];
   Process c or ch
}
```

Function Definitions

Return type *Parameter type and name*

```
double cube_volume(double side_length)
{
   double vol = side_length * side_length * side_length;
   return vol;
}
```

Exits function and returns result.

Reference parameter

```
void deposit(double& balance, double amount)
{
   balance = balance + amount;
}
```

Modifies supplied argument

Arrays

Element type *Length*

```
int numbers[5];
int squares[] = { 0, 1, 4, 9, 16 };
int magic_square[4][4] =
{
   { 16, 3, 2, 13 },
   { 5, 10, 11, 8 },
   { 9, 6, 7, 12 },
   { 4, 15, 14, 1 }
};

for (int i = 0; i < size; i++)
{
   Process numbers[i]
}
```

Vectors

Element type — *Initial values (C++ 11)*

```
#include<vector>

vector<int> values = { 0, 1, 4, 9, 16 };
```

Initially empty

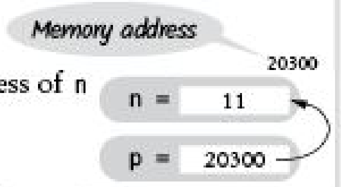
```
vector<string> names;
```

Add elements to the end

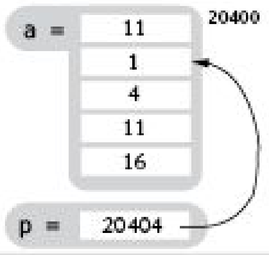
```
names.push_back("Ann");
names.push_back("Cindy"); // names.size() is now 2

names.pop_back(); // Removes last element

names[0] = "Beth"; // Use [] for element access
```

Pointers

```
int n = 10;
int* p = &n; // p set to address of n
*p = 11; // n is now 11
```



```
int a[5] = { 0, 1, 4, 9, 16 };
p = a; // p points to start of a
*p = 11; // a[0] is now 11
p++; // p points to a[1]
p[2] = 11; // a[3] is now 11
```



Input and Output

```
#include <iostream>
cin >> x; // x can be int, double, string
cout << x;
while (cin >> x) { Process x }
if (cin.fail()) // Previous input failed
#include <fstream>
string filename = ...;
ifstream in(filename);
ofstream out("output.txt");
string line; getline(in, line);
char ch; in.get(ch);
```

```
void increment_print() {
  static int s_value = 0; //static duration
  s_value++;
  cout << s_value << '\n';
} //s_value is not destroyed, but goes out of scope
int main() {
  increment_print(); //1
  increment_print(); //2
}
```

Static Variables

```
class Item {
private:
   int m_id;
   static int s_id_counter;
public:
   Item() {
      m_id = s_id_counter++;
   }
   int get_id() const {
      return m_id;
   }
};
int Item::s_id_counter = 1;
int main() { //
   Item first;
   Item second;
   cout << first.get_id();   //1
   cout << second.get_id();//2
}
```

Static Data Members

Range-based for Loop

An array, vector, or other container (C++ 11)

```
for (int v : values)
{
   cout << v << endl;
}
```

Output Manipulators

```
#include <iomanip>
```

`endl`	Output new line
`fixed`	Fixed format for floating-point
`setprecision(n)`	Number of digits after decimal point for fixed format
`setw(n)`	Field width for the next item
`left`	Left alignment (use for strings)
`right`	Right alignment (default)
`setfill(ch)`	Fill character (default: space)

Enumerations, Switch Statement

```
enum Color { RED, GREEN, BLUE };
Color my_color = RED;

switch (my_color) {
  case RED :
    cout << "red";  break;
  case GREEN:
    cout << "green"; break;
  case BLUE :
    cout << "blue"; break;
}
```

Class Definition

```
class BankAccount
{
public:
   BankAccount(double amount);     — Constructor declaration
   void deposit(double amount);    — Member function declaration
   double get_balance() const;     — Accessor member function
   ...
private:                           — Data member
   double balance;
};

void BankAccount::deposit(double amount)   ┐
{                                          ├ Member function definition
   balance = balance + amount;             │
}                                          ┘
```

Inheritance

Derived class — *Base class*

```
class CheckingAccount : public BankAccount
{
public:
   void deposit(double amount);    — Member function overrides base class
private:
   int transactions;               — Added data member in derived class
};

void CheckingAccount::deposit(double amount)
{
   BankAccount::deposit(amount);   — Calls base class member function
   transactions++;
}
```