Final Exam S25 Final V2

CSCI 13500: Software Analysis and Design 1

Hunter College, City University of New York

May 21, 2025, 11:30 AM - 1:30 PM, N118

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.
- Unless the problem explicitly requests, no need to include libraries and using namespace std.
- **Do not open this exam until instructed to do so.**

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 groceries[] = {"cake mix", "grape juice", "apple pie"}`, what is the value of groceries[0].substr(2, 5)?

(2) Given a declaration `std::vector<int> v(2, 1); v.push_back(0);`, what is the value of v.size()?

(3) What is the **minimum** integer that expression rand() % 7 -1 can generate?

(4) Given Given int `num = std::to_string(135).size() -2;`, where `to_string` converts an integer to a string and `size` method returns the number of characters of a string. What is the value for num?

(5) What is the value of `8 / (1 + 2) % 3` in C++?

(6) Write **header** of a function called hasEmptyStr, given an array *arr* of string type with *size* many elements, return whether the array has at least an empty string or not. If yes, return true, otherwise, return false.

(7) Declare class Time as follows.

```
1 class Time {
2 public:
3     int hour;
4     int minute;
5 };
```

Declare a Time object `curr` and initialize its hour as 8 and minute as 26.

(8) Given `int grades[] = {67, 92, 62};` What is the value of `*grades + 1`?

(9) Given the following code segment.

```
1  //foo works with array pf of int type with size many elements
2  void foo(int *pf, int size);
3
4  int main() {
5      int *arr = new int[20];
6
7      //TODO: write a statement to call foo for dynamically allocated array arr and
           its size.
8      //WRITE YOUR ANSWER IN THE FOLLOWING BOX.
9
10
11
12     delete[] arr;
13     arr = nullptr;
14
15     return 0;
16 }
```

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

```
1  int main() {
2     double a = 1.6;
3     int b = foo(&a, 't');
4     return 0;
5  }
```

What is the **header** of function foo?

(11) What is output for the following code?

```
string s = "12";
string *p = &s;
*p += "ab";
cout << s << endl;
```

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

```
vector<int> nums = {-2, 0, -1, 2, -5};

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

cout << count << endl;
```

(13) What the output of the following code?

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

int main() {
    for (int row = 0; row < 4; row++) {
        for (int col = 0; col < 3; col++) {
            if (col % 2 != 0)
               cout << "*";
            else cout << "#";
        }
        cout << endl;
    }
    return 0;
}
```

(14) What is the output of the following code? Assume that libraries and standard namespace are set up.

```
void foo(vector<string>& v);

int main() {
    vector<string> v = {"hey", "hi", "hello"};
    foo(v);

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

    return 0;
}

void foo(vector<string>& v) {
    int i = 0;
    int j = v.size() - 1;
    while (i < j) {
        swap(v[i], v[j]);
        i++;
        j--;
    }
}
```

(15) Given the following code, fill in the TODO part.

```
class Coord2D {
public:
    double x; //x-coordinate
    double y; //y-coordinate
};

double foo(Coor2D point) {
    //TODO: return the product of x- and y-coordinates of point
    //WRITE YOUR CODE IN THE FOLLOWING BOX.
}
```

2 (15 points) Answer the following questions.

(1) Define a function, `digit_space_only`, for a given string s, if it is **non-empty** and contains **only** digits and spaces, return true, otherwise, return false.

For example, `digit_space_only("")` returns false since it is an empty string.

`digit_space_only("12 3")` returns true.

`digit_space_only("12A b")` returns false since 'A' is not a digit or a space.

Hint: you may use the following functions from `cctype` library.

int isdigit (int c); Check if character is digit or not

int isspace (int c); Check if character is a whitespace or not

(2) Write a function `pointerToMin` that returns a **pointer** to the **first** appearance (if there are more than one occurrence) of the minimum value of an array of double type with *size* many elements.

If size is 0, return nullptr.

For example, suppose an array has elements 1.1, 3.3, 2.2, 3.3, 1.1, then the return of the function is a pointer to the first element.

Hint: you may use an index to the minimum element. Then use index and array name to get the pointer.

3 (10 points) Programming exercise on class

1. Define class for representing length in feet and inches. It is reasonable to define it to have two integer fields:

   `foot` for the number of feet, and

   `inch` for the number of inches. Note that a foot has 12 inches, so we need to make sure that inch is in [0, 11].

   Declare class `Length` with public data members `foot` and `inch`, both of `int` type.

Define non-member function `subtract`, given Length objects len and len2, the function should create and return a length object that is the result of subtracting `len2` from `len`. Example:

```
subtract({4, 6}, {2, 8}) // should return {1, 10}
```

Reason: 4 feet and 6 inches is 4 * 12 + 6 = 54 inches. And 2 feet 8 inches is 2 * 12 + 8 = 32 inches. Then 54 - 32 = 22 inches, which equals 1 feet and 10 inches.

Hint: For simplicity, we assume that `len` is no shorter than `len2`.

4 (10 points) Write codes of vector

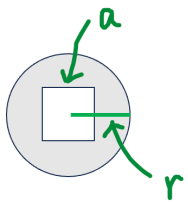
Define a function called `choose`, for a vector `v` of strings and a character (type char) `ch`, return a vector with all the elements from `v` whose strings **ending** with ch, in the same order. String `s` **ends** with character `ch` means `ch` is the **last** character of `s`.

For example, given a vector of strings with elements `"apple", "banana", "", "CDE", "orange"` and character 'e', the return is a vector with elements `"apple", "orange"`. Note that C++ is a case sensitive language, so 'e' is different from 'E'.

Hint: you may need to consider the case when a string is empty.

5 (15 points) Define class.

1. Define a SqCir as the region between a square nested into a circle. The shapes are concentric (share the same center). It has two parameters:



   (a) edge of the square **a**

   (b) radius of the circle **r**

2. **Assume that SqCir.hpp is provided** where data members **a** and **r** are declared as double types. **Your job** is to define **SqCir.cpp** with the following requirement.

3. Define a default constructor, set data members **a** to be 1 and **r** to be 2.

4. Define a non-default constructor, which takes formal parameters a and r, both are double types.

   (a) If both a and r are positive and $\sqrt{2}$a is smaller than or equal to 2r, set data member **a** by given parameter a and set data member **r** by given parameter r.

   (b) otherwise, set data members **a** to be 1 and **r** to be 2.

5. Define method **getArea**, return the value of $\pi r^2 - a^2$, where π is defined as `M_PI` in `cmath` library. Note that a and r are data members.

6. Define method **getPerimeter**, which returns $4a + 2\pi r$. Note that a and r are data members.

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

7. Create a SqCir object named **shape** from its non-default constructor with the edge of the square as 1 and the radius of the circle as 2.5.

8. Find out and print the area of **shape**.

9. Find out and print the perimeter of **shape**.

6 (10 points) function on vectors

Define a function called `fourOrMoreSucc`, given a vector of chars `v` and a char `toAdd`, do the following:

(1) Push `toAdd` to the back of `v` using `push_back` method of vector.

(2) Test whether there were 4 or more **consecutive** elements in the **back** of the vector. If so, return true, otherwise, return false.

For example, if the vector has elements `{'r', 'b', 'r'}`, and the element to add is `'r'`, then the return is false. Reason: after pushing `'r'` to the back of the vector, the elements change to `{'r', 'b', 'r', 'r'}`, but there are only two **consecutive** `'r'` in the **back** of the vector.

If the vector has elements `{'r', 'b', 'r', 'r', 'r'}`, and the element to add is `'r'`, then the return is true. Reason: after pushing `'r'` to the back of the vector, the elements change to `{'r', 'b', 'r', 'r', 'r', 'r'}`, and there are four **consecutive** `'r'` in the **back** of the vector.

7 (10 points) Define recursive function

Define a recursive function `reverse`, given an array of **double** with size many elements, reverse its elements. That is, swap the first and the last elements, swap the second and second to last elements, and so on. The return type is `void`.

For example, if an array with elements 1.1, 2.2, and 3.3, after the reverse, the array becomes 3.3, 2.2, 1.1

Warning: If you do not use recursion, you will not get any point.
No repetition statement, global or static variables are allowed in this function.
Use array, not vector.

Variable and Constant Definitions

```
int cans_per_pack = 6;

const double CAN_VOLUME = 0.335;
```

Type, *Name*, *Initial value*

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

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

vector<string> names;

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
```

Element type

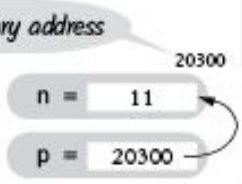
Initial values (C++ 11)

Initially empty

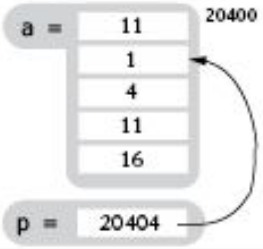
Add elements to the end

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);
```

Static Variables

```
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 Data Members

```
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
}
```

Range-based for Loop

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

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

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);
   void deposit(double amount);
   double get_balance() const;
   ...
private:
   double balance;
};

void BankAccount::deposit(double amount)
{
   balance = balance + amount;
}
```

Constructor declaration

Member function declaration

Accessor member function

Data member

Member function definition

Inheritance

```
class CheckingAccount : public BankAccount
{
public:
   void deposit(double amount);
private:
   int transactions;
};

void CheckingAccount::deposit(double amount)
{
   BankAccount::deposit(amount);
   transactions++;
}
```

Derived class

Base class

Member function overrides base class

Added data member in derived class

Calls base class member function