Row:	Seat:

Final Exam F23 V2
CSCI 13500: Software Analysis and Design 1
Hunter College, City University of New York

December 14, 2023, 9:00 - 11:00 AM, North Building 118

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.**

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 greetings[] = {"Hello", "Hi", "nice to meet you"}`, what is greetings[2][1]?

(2) Given Employee class, declare that class Doctor as subclass of Employee class with public inheritance.

(3) Write code to generate a random int between 10 and 20, where both ends are included. No library is needed.

(4) Given `string greeting = "Hello";` What is the value for greeting.substr(1,2)?

(5) Write a command to compile and link TestField.cpp and Field.cpp to generate a runnable file **run**.

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

(7) Write **header** of a function called stdev to return the standard deviation of an array of double numbers with size n.

(8) Given `int arr[] = {4, 3, 2, 1};` What is the value of `*arr + 1`?

(9) Declare and initialize a two-dimensional strings array called **synonyms** with three rows, each row with two columns. The first row is "kind", "nice", the second row is "big", "large", the third row is "small", "tiny".

(10) What is output for the following code?

```
vector<int> nums;
for (int i = 12; i >= 0; i--)
    nums.push_back(i);

for (int i = 0; i < nums.size(); i++)
    if (i % 4 == 0)
       cout << nums[i] << " ";

cout << endl;
```

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

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

int main() {
    int result = 0;
    for (int num = 6; num < 11; num += 4)
        result += num;

    cout << result << endl;
    return 0;
}
```

(12) What is output for the following code?

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

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

```
void foo(int& a);

int main() {
    int num = 1;
    foo(num);
    cout << num << endl;
    return 0;
}

void foo(int& a) {
    if ( a % 2 != 0 )
       a += 2;
    else a++;
}
```

(14) What the output when input is 75.1?

```
cout << "Enter a number: ";
double num;
cin >> num;
switch ((int)num / 10) {
    case 10:
    case 9:  cout << "excellent" << endl;
             break;
    case 8: cout << "good" << endl;
             break;
    case 7: cout << "ok" << endl;
             break;
    case 6: cout << "work hard" << endl;
             break;
    default: cout << "do not give up" << endl;
}
```

(15) What is the panel like when press up in game 1024? The empty cell is 0.

1		1
	1	1
1		1

2 (10 points) Answer the following questions.

(1) Define a function, for an given array of integers with its size, return number of elements that is positive.
For example, call the function with array with values -1, 0, -2, 0, 6, the size of array is 5, then the return is 1.

(2) Define function `void sortByLenRev(string* a, string* b)`, if the length of $*a$ is smaller than the length of $*b$, swap $*a$ with $*b$, otherwise, do nothing. Note that dereference operator * has lower precedence than dot operator.

3 (20 points) Programming exercises

(1) Define a function, for a given string, if it contains at least a letter **and** a special symbol in $, #,or !, return true, otherwise, return false.

For example, for string “abc”, the return is false. For string “#!”, the return is false. For “a!”, the return is true. For “!a”, the return is true.

Hint: you might use isalpha to check whether a charcter is a letter (alphabetic) or not.

int isalpha (int c); Check if character is alphabetic

You can count the number of occurrences of letters and number of occurrences of special symbols.

(2) Question on dynamically allocated memory

(a) Define **panel** to be int** type.

(b) Allocate memory of panel to be a two-dimensional array with 2 rows, each row has 3 columns.

(c) Initialize the element of panel indexed at (row)th row and (col)th column to be row + col, where row and col are indices and $0 <= row < 2$ and $0 <= col < 3$.

(d) Release the dynamically allocated memory and avoid dangling pointer problem.

4 (10 points) Write codes of vector

Define a function, for a given vector of strings, return a vector of all strings with odd length.

For example, call the above function on a vector of strings with values “ab”, “ccd”, “abcd”, the return is a vector of strings with value “ccd”.

5 (10 points) Define a class.

Here is Course.hpp of class **Course**.

```
#include <string>
class Course {
public:
    ...//omitted
private:
    std::string name; //represent course name
    int credit; //represent number of credit hour
};
```

Your job: define Course.cpp with the following requirement.

1. Include necessary library and header file.
2. Define a default constructor, which sets data member **name** to be "CS 127" and set data member **credit** to be 4.
3. Define a non-default constructor, which takes formal parameters name, a string, and credit, an int. Set data member **name** by given parameter name. If given parameter credit is positive, use it to set data member **credit**, otherwise, set data member **credit** to be 3.
4. Define method **getName** to return the value of data member **name**.

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
    virtual string toString() const; //return a textual information of name and age.
    ...//omit other constructors and methods
private:
    string name;
    int age;
};
```

Declare Student as a subclass of Person. Each student is a person, with additional data member **gpa**, which may contain decimal numbers. Suppose Person.hpp is properly declared. In Student.cpp, do the following:

Define non-default constructor of Student, which takes parameters name (a string), age (an int), and gpa (a double) to initialize the corresponding data members. This constructor can invoke the corresponding constructor of its super class, then initialize data member unique to the subclass. Data member gpa should be a double number in [0, 4]. If parameter gpa is not in [0, 4], set data member gpa to be 0.

Override toString method inherited from Person class to return a string representing the student's information like name, age, and gpa. You may use `string to_string (double val);` from std namespace to convert double number val to a string. Also, you can call toString method in the superclass.

7 (10 points) Define recursive function

Define a recursive function to check whether an array of ints is palindrome or not. An array of ints is palindrome if the elements read from left to right and from right to left are the same.

For example, array with values 1, 2, 1 is palindrome, but array with values 1, 2 is not palindrome.

Hint: an array is a palindrome if and only the leftmost element equals the rightmost element and the subarray from the second element to the second-to-last element is palindrome. Think what are the initial address and size of that 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

```
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

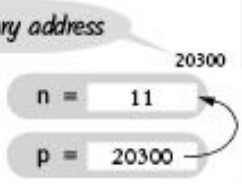
Initial values (C++ 11)

Initially empty

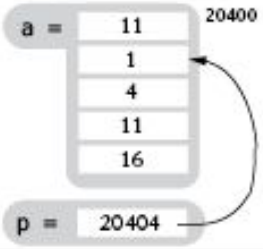
Add elements to the end

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