Answer:

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]?

Answer: greetings[2][1] is 'i'. Explanation: greetings[2] is the third element of array of strings, which is "nice to meet you". Expression greetings[2][1] is the second letter of this string, which is letter 'i'.

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

Answer: `class Doctor : public Employee`

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

Answer: Answer: Use rand() % 11 + 10 to generate a random int in [10, 20].

Explanation:

- There are $20 - 10 + 1 = 11$ integers from 10 to 20.
- `rand() % 11` generates a random integer from 0 to 10.
- rand() % 11 + 10 generates a random integer from 10 to 20.

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

Answer: the answer is string `"el"`.

Explanation: the first parameter is the starting index of the substring, number 1 is the index of the second letter.

The second prarameter in substr method is the length – aka, number of letters – in the substring.

So, greeting.substr(1, 2) means to extract a substring from string `greeting` whose value is "Hello". Starting from the letter indexed at 1 (the second letter), get 2 more letters, the result is `"el"`.

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

Answer: `g++ -o run TestField.cpp Field.cpp`

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

Answer: 3

Explanation: multiplication operator * and division operator / have the same precedence, and both are running from left to right. So * runs first in $5 * 2/3$, product of 5 * 2 is 10, after it is divided by 3, the result of integer division of 10 divided by 3 is like divide 10 pens among 3 students, each student gets 3 pens. So the result of $5 * 2/3$ is 3.

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

Answer: Use one the following:

```
double stdev(double* arr, int n);
```

```
double stdev(double arr[], int n);
```

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

Answer: 5

Explanation: dereference operator * has higher precedance than +. So we run *arr first.

- arr is the address of arr[0], so *arr is the element residing in that address, that is, arr[0], the first element of array `arr`.
- *arr + 1 adds 1 to arr[0]. So the result is 5.

(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".

Answer: Either one of the following will work. The key is the capacity for the second or higher dimension must be specified.

```
string synonyms[3][2] = { {"kind", "nice"}, {"big", "large"}, {"small", "tiny"} };
```

or

```
string synonyms[][2] = { {"kind", "nice"}, {"big", "large"}, {"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;
```

Answer: 12 8 4 0

Explanation: first put 12, $\cdots$, 0 to vector `nums`. Then

index i	0	1	2	3	4	5	6	7	8	9	10	11	12
nums[i]	12	11	10	9	8	7	6	5	4	3	2	1	0

Print out the elements whose index is a division of 4. So the print out is

12 8 4 0

(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;
}
```

Answer: 6 + 10 = 16

Explanation: num starts from 6, as long as it is smaller than 11, add it to variable result. After each round, increased num by 4.

The numbers are elegible to be added are 6 and 10. So result is 16.

You can also think in tabular format.

int result = 0;

num	$num < 11$?	result += num	num += 4
6	yes	result is increased by 6, changes from 0 to 6	num is increased by 4, and num changes to 10
10	yes	result is is increased by 10, changes from 6 to 16	num is increased by 4, and num changes to 14
14	no, stop		

(12) What is output for the following code?

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

Answer: 8

Explanation: after `int* p = &a`, which saves a's address to pointer p, then $*p$ represents the guy who lives in the address of variable a. Note that no two variables can reside in the same address, so *p is an alias of variable a.

So $*p+ = 6$; is the same as $a+ = 6$; Hence print out 8.

(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++;
}
```

Answer: 3

Symbol & after int function header means this parameter is passed by reference, ie, the original copy of actual parameter `num` is passed to formal parameter `a`. So, `a` is actually `num`.

Inside function body of foo, `a` – the original copy of `num` – is 1 and cannot be divided by 2, so `a` is increased by 2.

Since `a` is also an original copy of `num`, the change applies to `num`.

When function foo finishes and return to its caller, num keeps its its value 3.

(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;
}
```

Answer: ok

Explanation: (int)num changes a number to an int by truncating the decimal numbers. When num is 75.1, (int)num is 75. The result of integer division of 75 / 10 is 7, it is like to divide 75 pens among 10 kids, each kid gets 7 pens.

Then find out the label matching 7 and run the statements until statement break; or the end of switch statement is reached.

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

1		1
	1	1
1		1

Answer: After merging, the result looks like.

2	1	2
		1

Some students might select a random cell to place 1, that is ok.

Some students might not draw a framed table, and write the numbers in rows and columns and write 0 for empty cell, that is fine as well.

2 1 2
0 0 1
0 0 0

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.

Answer:

```
int getNumPositives(int arr[], int size) {
    int numPositives = 0;
    for (int i = 0; i < size; i++)
        if (arr[i] > 0)
           numPositives++;

    return numPositives;
}
```

A complete code is as follows.

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

int getNumPositives(int arr[], int size);
//int getNumPositives(int* arr, int size); //is also fine

int main() {
    int arr[] = {-1, 0, -2, 0, 6};
    int size = sizeof(arr) / sizeof(arr[0]);

    cout << getNumPositives(arr, size) << endl; //print 1
    return 0;
}

int getNumPositives(int arr[], int size) {
    int numPositives = 0;
    for (int i = 0; i < size; i++)
        if (arr[i] > 0)
           numPositives++;

    return numPositives;
}
```

(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.

Answer:

```
void sortByLenRev(string* a, string* b) {
    if ((*a).size() < (*b).size()) //ok
    //if (a->size() < b->size()) //use a->size() instead of (*a).size(), also ok
       swap(*a, *b);
}
```

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

void sortByLenRev(string* a, string* b);

int main() {
    string s1 = "hi";
    string s2 = "hello";

    sortByLenRev(&s1, &s2);

    cout << "s1 = " << s1 << ", s2 = " << s2 << endl;
    //print s1 = hello, s2 = hi
    return 0;
}

void sortByLenRev(string* a, string* b) {
    if ((*a).size() < (*b).size()) //ok
    //if (a->size() < b->size()) //use a->size() instead of (*a).size(), also ok
       swap(*a, *b);
}
```

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.

Answer:

```
bool hasLetterSpecialSymbol(string s) {
    int numLetters = 0;
    int numSpecialSymbols = 0;
    for (int i = 0; i < s.size(); i++)
        if (isalpha(s[i])) //can write as
          //if (s[i] >= 'A' && s[i] <= 'Z' ||
          //    s[i] >= 'a' && s[i] <= 'z')
          numLetters++;
         else if (s[i] == '$' || s[i] == '#' || s[i] == '!')
                numSpecialSymbols++;

     return numLetters > 0 && numSpecialSymbols > 0;
}
```

A complete code is as follows.

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

bool hasLetterSpecialSymbol(string s);

int main() {
    string strs[] = {"abc", "#!", "a!", "!a"};
    int size = sizeof(strs) / sizeof(strs[0]);

    for (int i = 0; i < size; i++)
        cout << boolalpha << hasLetterSpecialSymbol(strs[i]) << endl;
        //print false false true true, one in a row

    return 0;
```

```
}

bool hasLetterSpecialSymbol(string s) {
    int numLetters = 0;
    int numSpecialSymbols = 0;
    for (int i = 0; i < s.size(); i++)
        if (isalpha(s[i])) //can write as
          //if (s[i] >= 'A' && s[i] <= 'Z' ||
          //    s[i] >= 'a' && s[i] <= 'z')
          numLetters++;
         else if (s[i] == '$' || s[i] == '#' || s[i] == '!')
                numSpecialSymbols++;

     return numLetters > 0 && numSpecialSymbols > 0;
}
```

(2) Question on dynamically allocated memory

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

Answer: int** panel;

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

Answer:

```
int numRows = 2;
int numCols = 3;
panel = new int*[numRows];
for (int row = 0; row < numRows; row++)
    panel[row] = new int[numCols];
}
```

(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 $0 <= col < 3$.

Answer:

```
for (int row = 0; row < numRows; row++) //numRows can be written as 2
    for (int col = 0; col < numCols; col++) //numCols can be written as 3
        panel[row][col] = row + col;
```

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

Answer:

```
for (int row = 0; row < numRows; row++) { //numRows can be written as 2
    delete[] panel[row];
    panel[row] = nullptr;
}

delete[] panel;
panel = nullptr;
```

A complete code is as follows.

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

int main() {
    int** panel;

    int numRows = 2;
    int numCols = 3;
    panel = new int*[numRows];
    for (int row = 0; row < numRows; row++)
        panel[row] = new int[numCols];

    //set panel[row][col] by row + col
```

```
    for (int row = 0; row < numRows; row++) //numRows can be written as 2
        for (int col = 0; col < numCols; col++) //numCols can be written as 3
            panel[row][col] = row + col;

    //Release the dynamically allocated memory and avoid dangling pointer problem.
    for (int row = 0; row < numRows; row++) { //numRows can be written as 2
        delete[] panel[row];
        panel[row] = nullptr;
    }

    delete[] panel;
    panel = nullptr;

    return 0;
}
```

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”.

Answer:

```
vector<string> oddLen(vector<string> vec) {
    vector<string> result;

    for (int i = 0; i < vec.size(); i++)
        if (vec[i].size() % 2 != 0) //vec[i].size() can be replaced by vec[i].length()
          result.push_back(vec[i]);

    return result;
```

A complete code is shown as follows.

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

vector<string> oddLen(vector<string> vec);

int main() {
    vector<string> vec = {"ab", "ccd", "abcd"};

    vector<string> result = oddLen(vec);

    for (int i = 0; i < result.size(); i++)
        cout << result[i] << endl;
        //print
        //ccd

    return 0;
}

vector<string> oddLen(vector<string> vec) {
    vector<string> result;

    for (int i = 0; i < vec.size(); i++)
        if (vec[i].size() % 2 != 0) //vec[i].size() can be replaced by vec[i].length()
          result.push_back(vec[i]);

    return result;
}
```

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**.

Answer:

```
#include "Course.hpp"
#include <iostream>
#include <string> //need in some C++ versions
using namespace std;

Course::Course() {
    name = "CS 127";
    credit = 4;
}

Course::Course(string name, int credit) {
    this->name = name;
    if (credit > 0)
       this->credit = credit;
    else this->credit = 3;
}

string Course::getName() const {
    return name;
}
```

A complete code is as follows.
code of Course.hpp

```cpp
#ifndef Course_H
#define Course_H
#include <string> //need in some C++ version
class Course {
public:
    Course();
    Course(std::string name, int credit);
    std::string getName() const;
    int getCredit() const;
    void setName(std::string name);
    void setCredit(int credit);
private:
    std::string name; //it is discouraged to use namespace in hpp,
        //it might affect all the source code including this header file
    int credit;
};
#endif
```

code of Course.cpp

```cpp
#include "Course.hpp"
#include <iostream>
#include <string> //need in some C++ versions
using namespace std;

Course::Course() {
    name = "CS 127";
    credit = 4;
}

Course::Course(string name, int credit) {
    this->name = name;
    if (credit > 0)
       this->credit = credit;
    else this->credit = 3;
}

string Course::getName() const {
    return name;
}

int Course::getCredit() const {
    return credit;
}

void Course::setName(string name) {
    this->name = name;
}

```

```
void Course::setCredit(int credit) {
    if (credit > 0)
       this->credit = credit;
}
```

content of TestCourse.cpp

```
#include <iostream>
#include "Course.hpp"

using namespace std;

int main() {
    Course cs;

    cout << "name: " << cs.getName() << endl;
    cout << "credit: " << cs.getCredit() << endl;

    cs.setName("CS 235");
    cs.setCredit(3);

    cout << "name: " << cs.getName() << endl;
    cout << "credit: " << cs.getCredit() << endl;
    return 0;
}
```

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.

Answer:

```
Student::Student(string name, int age, double gpa) : Person(name, age) {
     if (gpa < 0 || gpa > 4)
        this->gpa = 0;
     else this->gpa = gpa;
}

string Student::toString() const {
    string str = Person::toString();
    str += "gpa: " + to_string(gpa) + "\n";
    //to_string(double) belongs to std namespace
    return str;
}
```

(optional) A complete code is as follows.
code of Person.hpp

```
#ifndef Person_H
#define Person_H
#include <string> //needed

//we normally do not add using namespace std;
//in a header file (ended with .hpp),
//since the source code that include
//the header file may not like to use that namespace.

class Person {
```

```
public:
    Person();
    Person(std::string name, int age);
    std::string getName() const;
    int getAge() const;
    void setAge(int age);
    void setName(std::string name);
    virtual std::string toString() const;

private:
    std::string name;
    int age;
};
#endif
```

code of Person.cpp

```
#include <iostream>
#include <string>
#include "Person.hpp"
using namespace std;

Person::Person() {
    name = "John Doe";
    age = 18;
}

Person::Person(string name, int age) {
    this->name = name;
    if (age >= 0 && age <= 130)
       this->age = age;
    else this->age = 18;
}

string Person::getName() const {
    return name;
}

int Person::getAge() const {
    return age;
}

void Person::setName(string name) {
    this->name = name;
}

void Person::setAge(int age) {
     if (age >= 0 && age <= 130)
        this->age = age;
}
```

```
34
35 string Person::toString() const {
36     string str = "";
37     str += "name: " + name + "\n";
38     str += "age: " + to_string(age) + "\n";
39     return str;
40 }
```

code of Student.hpp

```
1  #ifndef Student_H
2  #define Student_H
3  #include <string>
4  #include "Person.hpp"
5
6  class Student : public Person {
7  public:
8      Student();
9      Student(std::string name, int age, double gpa);
10     double getGpa() const;
11     void setGpa(double gpa);
12     virtual std::string toString() const;
13 private:
14     double gpa;
15 };
16 #endif
```

code of Student.cpp

```
1  #include "Student.hpp"
2  #include <string>
3  using namespace std;
4
5  //invoke default construtor of Person
6  Student::Student() : Person() {
7      gpa = 0;
8  }
9
10 Student::Student(string name, int age, double gpa) : Person(name, age) {
11      if (gpa < 0 || gpa > 4)
12         this->gpa = 0;
13      else this->gpa = gpa;
14 }
15
16 double Student::getGpa() const {
17     return gpa;
18 }
19
20 void Student::setGpa(double gpa) {
21     if (gpa >= 0 && gpa <= 4)
22        this->gpa = gpa;
```

```
}

string Student::toString() const {
    string str = Person::toString();
    str += "gpa: " + to_string(gpa) + "\n";
    //to_string(double) belongs to std namespace
    return str;
}
```

code of TestPersonStudent.cpp

```
#include <iostream>
#include "Person.hpp"
#include "Student.hpp"
using namespace std;
//sample output:
//name: Ann
//age: 27
//
//name: Bob
//age: 17
//gpa: 3.500000
//
//name: John Doe
//age: 18
//gpa: 0.000000

int main() {
    int size = 3;
    Person** personPtr = new Person*[size];
    //use Person* to store each information of a Person/Student
    //to avoid information of Student from slicing

    Person ann("Ann", 27);
    personPtr[0] = &ann; //personPtr[0] is Person* type

    Student bob("Bob", 17, 3.5);
       //call non-default constructor of Student
    personPtr[1] = &bob;

    Student johnDoe; //call default constructor of Student
    personPtr[2] = &johnDoe;

    for (int i = 0; i < size; i++)
        cout << personPtr[i]->toString() << endl;

    delete[] personPtr;
    personPtr = nullptr;
    return 0;
}
```

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.

Answer:

```
bool isPalindrome(int* arr, int size) {
    if (size <= 1)
       return true;

    return arr[0] == arr[size-1] && isPalindrome(arr+1, size-2);
}
```

A complete code is as follows.

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

bool isPalindrome(int* arr, int size);

int main() {
    int arr[] = {1, 2, 1};
    int size = sizeof(arr) / sizeof(arr[0]);

    cout << boolalpha << isPalindrome(arr, size) << endl; //true

    int arr2[] = {1, 2};
    int size2 = sizeof(arr2) / sizeof(arr2[0]);

    cout << boolalpha << isPalindrome(arr2, size2) << endl; //false

    int arr3[] = {1};
    int size3 = sizeof(arr3) / sizeof(arr3[0]);

    cout << boolalpha << isPalindrome(arr3, size3) << endl; //true
    return 0;
}

bool isPalindrome(int* arr, int size) {
    if (size <= 1)
       return true;

    return arr[0] == arr[size-1] && isPalindrome(arr+1, size-2);
}
```