**Answer:**

# Final Exam F24 Final V3

## CSCI 13500: Software Analysis and Design 1

## Hunter College, City University of New York

Dec 19, 2024, 1:45 PM - 3:45 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.
- **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[] = {"milk", "oat", "pecan pie"}`, what is groceries[2].substr(3, 4)?

**Answer:** groceries[2].substr(3, 4) is "an p". Explanation: groceries[2] is the third element of array of strings, which is "pecan pie". Expression groceries[2].substr(3, 4) is the substring from the fourth letter – index 3 – of this string spanning with 4 letters, which is substring "an p".

(2) Given a declaration `std::vector<int> v(10, 1);`, what is the value of v.size()?

**Answer:** `v.size()` returns the number of elements of v, which is 10 in this example.

(3) What possible numbers does code `rand() % 6 - 2` generate?

**Answer:** Answer: rand() % 6 generate a random int in [0, 5].

`rand() % 6 - 2` generates a random int in [-2, 3].

(4) Given `string numStr = std::to_string(10) + "25";`, where `to_string` converts an integer to a string. What is the value for numStr?

**Answer:** the answer is "1025".

(5) What is the value of `5 - 7 % 3 / 2` in C++?

**Answer:** 5

Explanation: division operator % has higher precedence than subtraction operator +. So % runs first in `7 % 3 / 2`. Note that 7 % 3 – the remainder of 7 pens divided among 3 persons – is 1. Next run 1 / 2, and the result is 0. Subtract 0 from 5, the result is 5.

(6) Write **header** of a function called sum, given an array of characters (type char) with *size* many elements, return the sum of ASCII code of all the elements in the array.

**Answer:** `int sum(char* arr, int size);` or `int sum(char arr[], int size);`

(7) Declare class Coord as follows.

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

Declare a Coord object `point` and initialize its x as 9 and y as 7.

**Answer:**

```
Coord point = {9, 7};
```

or

```
Coord point{9, 7};
```

or

```
Coord point;
point.x = 9;
point.y = 7;
```

(8) Given `int grades[] = {73, 100, 99, 62};` What is the value of `*grades + 3`?

**Answer:** 76

(9) Given the following code segment.

```
void foo(double *pf, double *pg);

int main() {
    double f = 1.7;
    double g = 2.0;

    //TODO: write a statement to call foo using appropriate attributes of f and g.

    return 0;
}
```

**Answer:** `foo(&f, &g)`

(10) Suppose we have main function defined as follows. And calling `foo(a, b, 3)`, the values of `a` is increased by 3 and `b` is decreased by 3. That is, `a` becomes ~~2~~ 4 and `b` becomes ~~5~~ -1.

```
int main() {
   int a = 1;
   int b = 2;
   foo(a, b, 3);
   return 0;
}
```

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

**Answer:** `void foo(int& a, int& b, int count);` //note that `&`'s after the first two parameters cannot be omitted and they means pass by reference.

(11) What is output for the following code?

```
    int a = 1;
    int* p = &a;
    a *= 2;
    cout << *p << endl;
```

**Answer:** 2

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 $a$.

a *= 2; is the same as $a = a * 2$;; so $a$ changes from the initital value 1 to 2. Then *p is 2.

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

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

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

    cout << count << endl;
```

**Answer:** 3

(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)
               cout << "*";
            else cout << "#";
        }
        cout << endl;
    }
    return 0;
}
```

**Answer:**

```
**#
**#
**#
**#
```

(14) What is panel after slide down operation?

| | | |
|---|---|---|
| 1 | 6 | 2 |
| 5 | | 8 |
| 3 | 4 | 7 |

**Answer:**

| | | |
|---|---|---|
| 1 | | 2 |
| 5 | 6 | 8 |
| 3 | 4 | 7 |

(15) Suppose in Project 3, data member `bins` have the following values,

`{{1, 1, 2}, {3, 2, 3}, {1, 2}, {3}}`,

After moving eligible element(s), according to rules listed in Project 3, from the second bin to the left to the rightmost bin, what are the elements in the **second bin to the left**?

**Answer:** 3, 2

# 2 (15 points) Answer the following questions.

(1) Define function `countSuccessiveEndElms`, for a given array of chars with its size, return the number of successive (aka consecutive) elements in the end of this array.

For example, call the function with array with values `'r', 'b', 'r', 'r'`, the size of array is 4. There are two `'r'`s residing successively (aka consecutively) in the end of array, the return is 2. Note that the leftmost `'r'` is not adjacent with those `'r'`s in the end, so it is not counted as part of the results.

**Answer:**

```
int countSuccessiveEndElms(char arr[], int size) {
    int i = size-1; //last index
    char ch = arr[i];
    int count = 0;
    while (i >= 0 && arr[i] == ch) {
        i--;
        count++;
    }

    return count;
}
```

In main function, write the following statements. No need to write the full definition of main function.

Define a char array with elements `'r', 'b', 'r', 'r'`.

Call the above function and print the number of successive end elements of the above array.

**Answer:**

```
    char arr[] = {'r', 'b', 'r', 'r'};
    int size = sizeof(arr) / sizeof(arr[0]);
    cout << countSuccessiveEndElms(arr, size) << endl; //2
```

**Answer:** A complete code to define and test the above function is as follows.

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

int countSuccessiveEndElms(char arr[], int size);

int main() {
    char arr[] = {'r', 'b', 'r', 'r'};
    int size = sizeof(arr) / sizeof(arr[0]);
    cout << countSuccessiveEndElms(arr, size) << endl; //2
    return 0;
}
```

```
13
14 int countSuccessiveEndElms(char arr[], int size) {
15     int i = size-1; //last index
16     char ch = arr[i];
17     int count = 0;
18     while (i >= 0 && arr[i] == ch) {
19         i--;
20         count++;
21     }
22
23     return count;
24 }
```

(2) Define function `searchFirst`, given an array of strings, its size, and a target (a string), return a pointer to the **first** occurrence of the target in an array, or nullptr if there is no match.

For example, suppose an array has elements `"how", "hi", "hello", "hi"`, if the target is `"hi"`, then the return of the function is a pointer to the second element. if the target is `"wonderful"`, then the return is nullptr.

**Answer:**

```
string* searchFirst(string arr[], int size, string target) {
    for (int i = 0; i < size; i++)
        if (arr[i] == target)
           return arr + i;

    return nullptr;
}
```

A complete code is as follows.

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

string* searchFirst(string arr[], int size, string target);

int main() {
    string arr[] = {"how", "hi", "hello", "hi"};
    int size = sizeof(arr) / sizeof(arr[0]);

    string* p = searchFirst(arr, size, "hi");

    cout << p << endl; //value depends on system and running time
    cout << *(p+1) << endl; //hey

    string *p2 = searchFirst(arr, size, "wonderful");
    cout << p2 << endl; //print 0x0

    return 0;
}

string* searchFirst(string arr[], int size, string target) {
    for (int i = 0; i < size; i++)
        if (arr[i] == target)
           return arr + i;

    return nullptr;
}
```

# 3 (10 points) Programming exercise on class

1. Define class for representing weight in pounds (also called lbs) and ounces. It is reasonable to define it to have two integer fields:

   lb for the number of pounds, and oz for the number of ounces. Note that a pound has 16 ounces, so we need to make sure that oz is in [0, 15].

```
class Weight {
public:
    int lb;
    int oz; //value in [0, 15]
};
```

   **Define** Weight addOzs(Weight curr, int ozVal);

   The function should create and return a weight object that is ozVal ounces more than curr. Note that 1 lb = 16 oz. Example:

   `addOzs({2, 8}, 23) // should return {3, 15}`

   Reason: 2 lbs 8 ounces is 2 * 16 + 8 = 40 ounces. Then 40 + 23 = 63 ounces, which equals 3 lbs and 15 ounces.

   **Answer:**

```
Weight addOzs(Weight curr, int ozVal) {
    int totalOzs = curr.lb * 16 + curr.oz;
    totalOzs += ozVal;
    Weight result;
    result.lb = totalOzs / 16;
    result.oz = totalOzs % 16;
    return result;
}
```

2. In main function, write the following statements. No need to define the whole main function.

   - Declare and instantiate `curr` as a Weight object with lb equals 2 and oz equals 8.
   - Declare and instanitate a Weight object called `heavier` that are 23 ounces more than `curr`. You may call `addOzs` with appropriate parameters.

   **Answer:**

```
    Weight curr = {2, 8};
    Weight heavier = addOzs(curr, 23);
    cout << heavier.lb << " pounds and " << heavier.oz << " ounces" << endl;
    //3 pounds and 15 ounces
```

**Answer:**

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

class Weight {
public:
    int lb;
    int oz;
};

Weight addOzs(Weight curr, int ozVal);

int main() {
    Weight curr = {2, 8};
    Weight heavier = addOzs(curr, 23);
    cout << heavier.lb << " pounds and " << heavier.oz << " ounces" << endl;
    //3 pounds and 15 ounces
    return 0;
}

Weight addOzs(Weight curr, int ozVal) {
    int totalOzs = curr.lb * 16 + curr.oz;
    totalOzs += ozVal;
    Weight result;
    result.lb = totalOzs / 16;
    result.oz = totalOzs % 16;
    return result;
}
```

# 4 (10 points) Write codes of vector

Define a function called `choose`, for a vector `v` of characters (type char), return a vector with all the elements from `v` that are uppercase letters, in the same order. In English, uppercase letters are `'A' - 'Z'`

For example, given a vector of characters with elements `'a', 'B', '#', 'D', 'c'`, the return is a vector with elements `'B', 'D'`.

Hint: int isupper ( int c ); checks if parameter c is an uppercase alphabetic letter. Return a value different from zero (i.e., true) if indeed c is an uppercase alphabetic letter. Zero (i.e., false) otherwise.

isupper is from cctype library. However, you do not need to include library in your code.

**Answer:**

```
vector<char> choose(vector<char> v) {
    vector<char> results;
    for (int i = 0; i < v.size(); i++) {
        if (isupper(v[i]))
          results.push_back(v[i]);
    }

    return results;
}
```

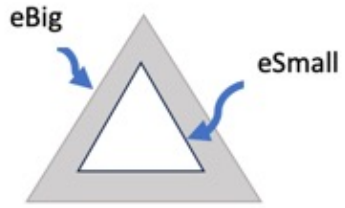
A complete code is shown as follows.

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

vector<char> choose(vector<char> v);

int main() {
    vector<char> v = {'a', 'B', '#', 'D', 'c'};

    vector<char> results = choose(v);

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

    cout << endl;

    return 0;
}

vector<char> choose(vector<char> v) {
    vector<char> results;
    for (int i = 0; i < v.size(); i++) {
```

```
25          if (isupper(v[i]))
26             results.push_back(v[i]);
27      }
28
29      return results;
30  }
```

# 5 (15 points) Define class for triangular ring shape.

1. Define a triangular ring as the region between two concentric equilateral triangles (aka the sides of a triangle are of same length). Call it a TriRing. It has two parameters:




   (a) edge of the inner or the smaller triangle **eSmall**

   (b) edge of the outer or the bigger triangle **eBig**

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

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

   **Answer:**

```
1  TriRing::TriRing() {
2      eSmall = 1;
3      eBig = 2;
4  }
```

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

   (a) If both eSmall and eBig are positive and eBig is larger than eSmall, set data member **eSmall** by given parameter eSmall and set data member **eBig** by given parameter eBig.

   (b) otherwise, set data members **eSmall** to be 1 and **eBig** to be 2.

   **Answer:**

```
1  TriRing::TriRing(double eSmall, double eBig) {
2      if (eSmall > 0 && eBig > 0 && eBig > eSmall) {
3         this->eSmall = eSmall;
4         this->eBig = eBig;
5      }
6      else {
7          this->eSmall = 1;
8          this->eBig = 2;
9      }
10 }
```

5. Define method **getArea**, return the value of $\frac{\sqrt{3}}{4}(eBig)^2$ - $\frac{\sqrt{3}}{4}(eSmall)^2$, where `sqrt` is defined as square root in `cmath` library. Note that $eBig$ and $eSmall$ are data members, not e * Big or e * Small.

**Answer:**

```
double TriRing::getArea() const {
    return sqrt(3) / 4 * eBig * eBig - sqrt(3) / 4 * eSmall * eSmall;
}
```

6. Define method **getPerimeter**, which returns $3(eSmall) + 3(eBig)$. Note that $eBig$ and $eSmall$ are data members, not e * Big or e * Small.

**Answer:**

```
double TriRing::getPerimeter() const {
    return 3 * eBig + 3 * eSmall;
}
```

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

7. Create a TriRing object named **tri** from its default constructor.

**Answer:**

```
    TriRing tri;
```

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

**Answer:**

```
    cout << "area: " << tri.getArea() << endl;
```

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

**Answer:**

```
    cout << "perimeter: " << tri.getPerimeter() << endl;
```

**Answer:** A complete code is as follows.
code of TriRing.hpp

```
#ifndef TRI_RING_H
#define TRI_RING_H
class TriRing {
public:
    TriRing();
    TriRing(double eSmall, double eBig);
    double getArea() const;
```

```
    double getPerimeter() const;

private:
    double eSmall; //edge of the inner or smaller triangle
    double eBig; //edge of the outer or bigger triangle
};
#endif
```

Code of TriRing.cpp

```
#include "TriRing.hpp"
#include <cmath>

TriRing::TriRing() {
    eSmall = 1;
    eBig = 2;
}

TriRing::TriRing(double eSmall, double eBig) {
    if (eSmall > 0 && eBig > 0 && eBig > eSmall) {
       this->eSmall = eSmall;
       this->eBig = eBig;
    }
    else {
        this->eSmall = 1;
        this->eBig = 2;
    }
}

double TriRing::getArea() const {
    return sqrt(3) / 4 * eBig * eBig - sqrt(3) / 4 * eSmall * eSmall;
}

double TriRing::getPerimeter() const {
    return 3 * eBig + 3 * eSmall;
}
```

code of TriRingTest.cpp

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

//sample output:
//area: 1.29904
//perimeter: 9
int main() {
```

```
10      TriRing tri;
11      cout << "area: " << tri.getArea() << endl;
12      cout << "perimeter: " << tri.getPerimeter() << endl;
13
14      return 0;
15  }
```

# 6 (10 points) function on vectors

Define a function called `compare`, given two vectors of strings, if they have the same number of elements, find out whether the length of **every** element in the first vector is larger than that of the same-index element in the second vector, if yes, return true, otherwise, return false. If these vectors do not have the same number of elements, return false.

For example, if the first vector is {"hello", "hi"} and the second vector is {"abcdef", "abc", "123"}, the return is false. Reason: the two vectors have different number of elements.

If the first vector is {"hellooo", "hey", "abcd"} and the second vector is {"hello", "hi", "how"}, return true. Reason: both vectors have the same number of elements. Furthermore, the length of the first element `"hellooo"` in the first vector is larger than the length of the first element `"hello"` in the second vector. The length of the second element `"hey"` in the first vector is larger than the length of the second element `"hi"` in the second vector. The length of the third element `"abcd"` in the first vector is larger than the length of the third element `"how"` in the second vector.

If the first vector is {"abcdef", "ab"} and the second vector is {"hello", "hi"}, the return is false. Reason: even though both vectors have the same number of elements, the length of the second element `"ab"` of the first vector is not larger than the length of the second element `"hi"` of the second vector.

**Answer:** function compare is defined as follows.

```
bool compare(vector<string> v1, vector<string> v2) {
    if (v1.size() != v2.size())
       return false;

    //Now v1.size() is the same as v2.size().
    for (int i = 0; i < v1.size(); i++)
        if (v1[i].length() >= v2[i].length())
           return false;

    return true;
}
```

A complete code is as follows.

```
//Given two vectors of strings, if they have the same number of elements, find out
   whether the length of every element in the first vector is larger than that of the
    corresponding element in the second vector, if yes, return true, otherwise,
   return false. If these vectors do not have the same number of elements, return
   false.

//For example, if the first vector is {"hello", "hi", "how"} and the second vector is
   {"hellooo", "hey", "abcd"}, return true.

//If the first vector is {"hello", "hi"} and the second vector is {"abcdef", "abc",
   "123"}, the return if false.

#include <iostream>
#include <string>
```

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

bool compare(vector<string> v1, vector<string> v2);

int main() {
    vector<string> v1 = {"hello", "hi", "how"};
    vector<string> v2 = {"helloo", "hey", "abcd"};

    cout << boolalpha << compare(v1, v2) << endl; //true

    vector<string> v3 = {"hello", "hi"};
    vector<string> v4 = {"abcdef", "abc", "123"};

    cout << boolalpha << compare(v3, v4) << endl; //false
    return 0;
}

bool compare(vector<string> v1, vector<string> v2) {
    if (v1.size() != v2.size())
       return false;

    //Now v1.size() is the same as v2.size().
    for (int i = 0; i < v1.size(); i++)
        if (v1[i].length() >= v2[i].length())
           return false;

    return true;
}
```

# 7 (10 points) Define recursive function

Define a recursive function `printArray`, given an array of int with size many elements, print all elements from the first one to the last one, separated by a space, in the same line.

For example, if an array with elements 1, 2, and 3, the print is
1 2 3

**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.**

**Answer:** Code of function is as follows.

```
void printArray(int* arr, int size) {
    if (size == 1) {
       cout << arr[0];
       return;
    }

    cout << arr[0];
    cout << " ";
    printArray(arr+1, size-1); //do not write size-1 as size, otherwise, there is a
        segment error
}
```

A complete code is as follows.

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

void printArray(int* arr, int size);

int main() {
    int arr[] = {1, 2, 3};
    int size = sizeof(arr) / sizeof(arr[0]);

    printArray(arr, size);
    return 0;
}

void printArray(int* arr, int size) {
    if (size == 1) {
       cout << arr[0];
       return;
    }

    cout << arr[0];
    cout << " ";
```

```
23      printArray(arr+1, size-1); //do not write size-1 as size, otherwise, there is a
            segment error
24  }
```