Answer:

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

1 (30 points) Answer the following questions.

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

Answer: animals[2].substr(5, 3) is "ant". Explanation: animals[2] is the third element of array of strings, which is "elephant". Expression animals[2].substr(5, 3) is the substring from the sixth letter of this string with 3 letters, which is substring "ant".

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

Answer: `class Shark : public Fish`

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

Answer: Answer: `rand() % 9 + 3` generate a random int 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`?

Answer: 1

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

Answer: `g++ -c Hare.cpp`

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

Answer: 3

(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.

Answer: `string concat(char* charArr, int size);` or `string concat(char charArr[], int size);`

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

Answer: 77

(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.

Answer:

```
1 double* p = new double[20];
```

or

```
1 double* p;
2 p = new double[20];
```

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

```
1 int main() {
2    int n = 2;
3    foo("hello", &n);
4    return 0;
5 }
```

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

Answer: void foo(string s, int* p); or void foo(string, int*);

The first parameter is a string.

&n is the address of n, which is an integer variable. So the second formal parameter of foo should be int*.

(11) What is output for the following code?

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

Answer: 8

Explanation: after `int* p = &a`, which saves a's address to pointer p, then $*p$ represents the variable whose ad
where p is 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 6 to 8. Then *p is 8.

(12) What the output of the following code for `foo(3)`?

```
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;
    }
}
```

Answer:

```
 *
***
```

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

```
#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;
```

```
}
```

Answer: The code return the number of characters in input that is neither 'a' nor 'b'.

2

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

```
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;
```

Answer: 6

(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;
}
```

Answer: 2 3 7

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.

   Answer:

```
//Note: you can also use int to replace size_t,
//but size_t is non-negative integer, so it is more appropriate.
double percentage(char arr[], size_t size) {
    int num = 0;
    for (int i = 0; i < size; i++)
        if (arr[i] == 'A' || arr[i] == 'B')
           num++;

    return 100.0 * num / size;
}
```

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

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

//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.

double percentage(char arr[], size_t size);

int main() {
    char arr[] = {'A', 'B', 'C', 'C', 'D', 'F'};
    int size = sizeof(arr) / sizeof(arr[0]);
    cout << percentage(arr, size) << endl;

    return 0;
}

double percentage(char arr[], size_t size) {
    int num = 0;
    for (int i = 0; i < size; i++)
        if (arr[i] == 'A' || arr[i] == 'B')
           num++;
```

```
23
24      return 100.0 * num / size;
25  }
```

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?

Answer:

```
Date add_month(Date curr, int num_months) {
    int newYear = curr.year + num_months / 12;
    int newMonth = curr.month + num_months % 12;
    if (newMonth > 12) {
       newMonth -= 12;
       newYear++;
    }

    Date newDate = {newYear, newMonth};
    return newDate;
}
```

A complete code is as follows.

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

class Date {
public:
    int year;
    int month;
};

Date add_month(Date curr, int num_months);
```

```
12
13 int main() {
14     Date curr = {2021, 12};
15     Date after = add_month(curr, 25);
16     cout << after.year << " " << after.month << endl;
17     return 0;
18 }
19
20 Date add_month(Date curr, int num_months) {
21     if (num_months <= 0)
22        return curr;
23
24     int newYear = curr.year + num_months / 12;
25     int newMonth = curr.month + num_months % 12;
26     if (newMonth > 12) {
27        newMonth -= 12;
28        newYear++;
29     }
30
31     Date newDate = {newYear, newMonth};
32     return newDate;
33 }
```

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.

Answer:

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

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

double slope(Coord2D* p1, Coord2D* p2);

int main() {
    Coord2D a = {1, 2};
    Coord2D b = {6, 3};
    Coord2D c = {1, 3};
```

```
17
18      cout << slope(&a, &b) << endl; //0.2
19      cout << slope(&a, &c) << endl; //inf
20
21      return 0;
22  }
23
24  double slope(Coord2D* p1, Coord2D* p2) {
25      if (p1->x == p2->x)
26          return std::numeric_limits<double>::infinity();
27
28      return (p1->y - p2->y) / (p1->x - p2->x);
29  }
```

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.

Answer:

```
vector<int> choose(vector<int> v, int left, int right) {
    vector<int> result;
    for (int i = 0; i < v.size(); i++) {
        if (v[i] >= left && v[i] <= right)
          result.push_back(v[i]);
    }

    return result;
}
```

A complete code is shown as follows.

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

vector<int> choose(vector<int> v, int left, int right);
int main() {
    vector<int> v = {12, 3, 6, 5, 7};
    vector<int> result = choose(v, 3, 6);

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

    return 0;
}

vector<int> choose(vector<int> v, int left, int right) {
    vector<int> result;
    for (int i = 0; i < v.size(); i++) {
        if (v[i] >= left && v[i] <= right)
          result.push_back(v[i]);
    }

    return result;
}
```

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.

   Answer:

```
Pentagon::Pentagon() {
    side = 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.

   Answer:

```
Pentagon::Pentagon(double side) {
    if (side > 0)
       this->side = side;
    else this->side = 1;
}
```

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

   Answer:

```
void Pentagon::setSide(double side) {
    if (side > 0)
       this->side = side;
}
```

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

   Answer:

```
double Pentagon::getPerimeter() const {
    return 5 * side;
}
```

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);

   Answer:

```
double Pentagon::getArea() const {
    return 1.0 / 4 * sqrt(5 * (5 + 2 * sqrt(5))) * side * side;
}
```

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

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

   Answer:

```
    Pentagon pente;
```

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

   Answer:

```
    cout << pente.getArea() << endl;
```

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

   Answer:

```
    pente.setSide(2);
```

Answer: A complete code of Pentagon is as follows.

```
#ifndef Pentagon_H
#define Pentagon_H

class Pentagon {
public:
    Pentagon();
    Pentagon(double side);
    double getArea() const;
    double getPerimeter() const;
    void setSide(double side);
private:
    double side;
};
#endif
```

code of Pentagon.cpp

```cpp
#include "Pentagon.hpp"
#include <cmath> //sqrt

Pentagon::Pentagon() {
    side = 1;
}

Pentagon::Pentagon(double side) {
    if (side > 0)
       this->side = side;
    else this->side = 1;
}

double Pentagon::getArea() const {
    return 1.0 / 4 * sqrt(5 * (5 + 2 * sqrt(5))) * side * side;
}

double Pentagon::getPerimeter() const {
    return 5 * side;
}

void Pentagon::setSide(double side) {
    if (side > 0)
       this->side = side;
}
```

content of PentagonTest.cpp

```cpp
#include <iostream>
#include <string>
#include "Pentagon.hpp"

using namespace std;

int main() {
    //Create a Pentagon object named pente from its default constructor.
    Pentagon pente;

    //Print out the area of pente.
    cout << pente.getArea() << endl;

    //Reset the side of pente to be 2.
    pente.setSide(2);
    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
    ...//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**.

Answer:

```
Student::Student(std::string name, int age, std::string major) : Person(name,
    age) {
    if (major != "")
       majors.push_back(major);
    else majors.push_back("undecided");
}
```

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

Answer:

```
int Student::getNumMajors() const {
   return majors.size();
}
```

Answer: (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;
}

string Person::toString() const {
    string str = "";
    str += "name: " + name + "\n";
    str += "age: " + to_string(age) + "\n";
    return str;
}
```

code of Student.hpp

```
#ifndef Student_H
#define Student_H
#include <vector>
#include <string>
#include "Person.hpp"

//(1) We normally do not use standard namespace in a header file,
//    ended by .hpp; otherwise, all the files included
//    the header file will have to use standard namespace as well.
//    This is like, English is the most popular language,
//    but we do not set it as default language in every setting.
//(2) Without using namespace std, we use std::vector instead of
//    vector, use std::string instead of string.
class Student : public Person {
public:
    Student(std::string name, int age, std::string major);
    int getNumMajors() const;
    std::string toString() const override; //override can be omitted, and can only be
       used in c++11 or above
    void addMoreMajor(std::string major);
private:
    std::vector<std::string> majors;
};
```

```
#endif
```

code of Student.cpp

```
#include "Student.hpp"

Student::Student(std::string name, int age, std::string major) : Person(name, age) {
    if (major != "")
       majors.push_back(major);
    else majors.push_back("undecided");
}

std::string Student::toString() const { //override cannot be here
    std::string str = Person::toString();
    str += "majors:\n"; //\n means new line
    for (int i = 0; i < majors.size(); i++)
        str += majors[i];

    return str;
}

//add more majors
void Student::addMoreMajor(std::string major) {
    if (major != "")
       majors.push_back(major);
}

int Student::getNumMajors() const {
   return majors.size();
}
```

Code of StudentTest.cpp

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

//Put Person.hpp, Person.cpp, Student.hpp, Student.cpp, and StudentTest.cpp in the
    same folder
//run the following commands
//g++ -std=c++11 *.cpp
//./a.out

//sample output:
//name: Ann
//age: 18
//majors:
```

```
//Computer Science

int main() {
    Student ann("Ann", 18, "Computer Science");
    cout << ann.toString() << endl;
    return 0;
}
```

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 integers 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.

Answer:

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

int getMax(int* arr, int size);

int main() {
    int arr[] = {2, 3, 1};
    int size = sizeof(arr) / sizeof(arr[0]);
    cout << getMax(arr, size) << endl;
    return 0;
}

int getMax(int* arr, int size) {
   if (size == 1)
      return arr[0];

   int value = getMax(arr+1, size-1);
   //find out the max in subarray from
   //the second element to the last element
   //return max(arr[0], value);
   //you can also replace the above return statement by
   if (arr[0] > value)
      return arr[0];
   else return value;
}
```