Answer:

Final Exam F25 Final V1
CSCI 13500: Software Analysis and Design 1
Hunter College, City University of New York

December 18, 2025, 9:00 AM - 11:00 AM, N 118

1 (30 points) Answer the following questions.

(1) Given `std::string colors[] = {"green", "light blue", "cyan"}`, what is colors[1].substr(3)?

Answer: colors[1].substr(3) is “ht blue”. Explanation: colors[1] is the second element of array of strings, which is “light blue”. Expression colors[1].substr(3) is the substring from the fourth letter – index 3 – of this string to the end, which is substring “ht blue”.

(2) Given a declaration `std::vector<int> v(10, 3); v.pop_back();`, what is the value of `v.size()`?

Answer: 9

explanation: `vector<int> v(10, 3);` creates a vector of integers with 10 elements, each element has value 3. After pop the last elements from it, v has one fewer element. Hence, there are 9 elements in vector v.

(3) How to generate a random integer in [100, 160]?

Answer: `rand() % 61 + 100`

`rand() % 61` generates a random int in [0, 60].

`rand() % 61 + 100` generates a random int in [100, 160].

(4) Given `std::string str = std::to_string(135) + "-01";`, where `to_string` converts an integer to a string. What is the value for `str`?

Answer: `"135-01"`.

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

Answer: 3

Explanation: expression in parentheses runs first. Run 2 * 4 % 5. It is like to divide 8 pens among 5 persons, each person get 1 pens, 3 pens left.

(6) Write **header** of a function called remove, given an array *arr* of **string** type with *size* many elements, remove the last element from the array. Return type of the function is `void`.

Answer: `void remove(std::string* arr, int& size);` or `void remove(std::string arr[], int& size);` or

`void remove(string* arr, int& size);` or

`void remove(string arr[], int& size);` or

(7) Declare class Coord3D as follows.

```cpp
class Coord3D {
public:
  double x;
  double y;
  double z;
};
```

Declare a Coord3D object `point` and initialize its x, y, z to be 1, 2, 3, respectively.

Answer:

```cpp
Coord3D point = {1, 2, 3};
```

or

```cpp
Coord3D point{1, 2, 3};
```

or

```cpp
Coord3D point;
point.x = 1;
point.y = 2;
point.z = 3;
```

(8) Given `int grades[] = {76, 100, 72, 96};` What is the value of `*(grades+1) - 1`?

Answer: 99

(9) Given the following code segment, finish the TODO part.

```cpp
int main() {
  std::cout << "Enter size of an array: ";
  int size;
  std::cin >> size;
  while (size <= 0) {
    std::cout << "Re-enter, the size is not positive: ";
    std::cin >> size;
  }

  //TODO: write a statement to declare and initialize a pointer arr to
  //a dynamically allocated array with size elements of double type.
  //WRITE YOUR ANSWER IN THE FOLLOWING BOX.


  //omit other code ...
  return 0;
}
```

Answer: `double* arr = new double[size];`

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

```
int main() {
  double m = 1.1;
  double n = 2.2;
  //In foo, if m < n, exchange the values of m and n.
  //The return type is void.
  foo(&m, &n);

  std::cout << m << " " << n << " " << std::endl;
  //expected output: 2.2 1.1
  return 0;
}
```

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

Answer: `void foo(double* p1, double* p2);`

A complete code is shown as follows.

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

void foo(double* p1, double* p2);

int main() {
  double m = 1.1;
  double n = 2.2;
  //In foo, if m < n, exchange the values of m and n.
  //The return type is void.
  foo(&m, &n);

  std::cout << m << " " << n << " " << std::endl;
  //expected output: 2.2 1.1
  return 0;
}

void foo(double* m, double* n) {
  if (*m < *n) {
    std::swap(*m, *n);
  }
}
```

(11) What is output calling `foo` with an array with elements -5, -6, 0, 7, 2 and its corresponding size?

```
int foo(int* arr, int size) {
  int i = 0;
  while (i < size && arr[i] <= 0) {
    i++;
  }

  return i;
}
```

Answer: 3

Explanation: the above loop is to return the index of the **first** positive element in the array; if every element of the array is non-positive, then return the number of elements of the array.

In this example, the first three elements are non-positive but not the fourth, which is indexed at 3. Note that the index of the first element is 0.

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

```
std::vector<int> nums = {1, -2, 2, -1};

int result = 0;
for (int i = 0; i < nums.size(); i++) {
  if (nums[i] > 0) {
    result += nums[i];
  }
}

std::cout << result << std::endl;
```

Answer: 3

Print out the sum of all positive integers in an int vector.

```
//Purpose: print out the sum of all positive integers in an int vector
#include <iostream>
#include <string>
#include <vector>

int main() {
  std::vector<int> nums = {1, -2, 2, -1};

  int result = 0;
  for (int i = 0; i < nums.size(); i++) {
    if (nums[i] > 0) {
      result += nums[i];
    }
  }

  std::cout << result << std::endl; //3

  return 0;
}
```

(13) Given arr with values 1, -2, 2, 6, 0 with size 5, what will be the value of arr after calling foo on arr and size?

```
void foo(int arr[], int size) {
  int i = 0;
  int j = size-1;
  while (i < j) {
    std::swap(arr[i], arr[j]);
    i++;
    j--;
  }
}
```

Answer: 0 6 2 -2 1

A complete code is as follows.

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

void foo(int arr[], int size);

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

  for (int i = 0; i < size; i++) {
    std::cout << arr[i] << " ";
  }
  std::cout << std::endl;

  return 0;
}

void foo(int arr[], int size) {
  int i = 0;
  int j = size-1;
  while (i < j) {
    std::swap(arr[i], arr[j]);
    i++;
    j--;
  }
}
```

(14) What the output of the following code? For simplicity, we omit libraries.

```
int main() {
  int size = 3;
  for (int row = 0; row < size; row++) {
    for (int col = 0; col < size; col++) {
      if (row + col < size) {
        std::cout << "*";
      }
      else {
        std::cout << "-";
      }
    }
    std::cout << std::endl;
  }

  return 0;
}
```

Answer:

```
***
**-
*--
```

A complete code is as follows.

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

int main() {
  int size = 3;
  for (int row = 0; row < size; row++) {
    for (int col = 0; col < size; col++) {
      if (row + col < size) {
        std::cout << "*";
      }
      else {
        std::cout << "-";
      }
    }
    std::cout << std::endl;
  }

  return 0;
}
```

(15) What is the output of the following code? Assume that libraries are set up.

```
std::string foo(std::vector<std::string> v, char ch);

int main() {
  std::vector<std::string> v = {"apple", "", "blue", "corn", "berry", "ant"};
  std::cout << foo(v, 'a') << std::endl;
  return 0;
}

std::string foo(std::vector<std::string> v, char ch) {
  std::string result;
  for (int i = 0; i < v.size(); i++) {
    if (v[i] != "" && v[i][0] == ch) {
      result += v[i] + ",";
    }
  }

  return result;
}
```

Answer: apple,ant,

Explanation: concatenate strings starting with letter ‘a’.

A complete code is as follows.

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

std::string foo(std::vector<std::string> v, char ch);

int main() {
  std::vector<std::string> v = {"apple", "", "blue", "corn", "berry", "ant"};
  std::cout << foo(v, 'a') << std::endl;
  return 0;
}

```

```
std::string foo(std::vector<std::string> v, char ch) {
  std::string result;
  for (int i = 0; i < v.size(); i++) {
    if (v[i] != "" && v[i][0] == ch) {
      result += v[i] + ",";
    }
  }

  return result;
}
```

2 (15 points) Answer the following questions.

(1) Define a function, `getInitial`, for a string in the format of first name first, followed by a space symbol (`' '`), followed by last name, return a string representing the initial.

For example, given a string with value `"George Washington"`, the returned string is `"GW"`.

Hint: extract the first name and the last name. Instantiate an empty string. Concatenate the empty string with the first letter of the first name, then the first letter of the last name. You might use the following methods from string class.

`size_t find (char c, size_t pos = 0) const;`

Searches the string for the first occurrence of the sequence specified by its arguments.

`string substr (size_t pos = 0, size_t len = npos) const;`

Returns a newly constructed string object with its value initialized to a copy of a substring of this object.

Answer:

```
//Purpose: for a name in the format of first name first and
//last name last, where first name and last name are separated by a space,
//return the initial of a name by concatenating the first letter of
//the first name and the first letter of the last name.
#include <iostream>
#include <string>

std::string getInitial(std::string name);

int main() {
  std::cout << getInitial("George Washington") << std::endl; //print "GW" without
      quotes
  return 0;
}

std::string getInitial(std::string name) {
  size_t spIdx = name.find(' ');
  std::string firstName = name.substr(0, spIdx);
  std::string lastName = name.substr(spIdx+1);

  std::string initial;
  initial += firstName[0];
  initial += lastName[0];

  return initial;
}
```

(2) Write a function `pointerToLastEven` that returns a **pointer** to the **last** occurrence (if there are more than one occurrence) of the even integer in an array of **int** type with *size* many elements.

If size is 0 or there is no even integer, return nullptr.

For example, given an array with elements 5, 4, 3, 2, 1, the return of the function is a pointer to element 2. Given an array with elements 1, 3, 5, the return is nullptr.

Answer: A solution is as follows.

```
int* pointerToLastEven(int arr[], int size) {
  for (int index = size-1; index >= 0; index--) {
    if (arr[index] % 2 == 0) {
      return arr + index;
    }
  }

  return nullptr;
}
```

A complete code is as follows.

```
//Purpose: return the pointer to the last even number in the array
#include <iostream>
#include <string>

int* pointerToLastEven(int arr[], int size);

int main() {
  int arr[] = {1, 2, 3, 4, 5};
  int size = sizeof(arr) / sizeof(arr[0]);

  int* ptr = pointerToLastEven(arr, size);
  std::cout << ptr << std::endl; //0x16dd3ed9c
  //the actual address changes from different systems and different running time
  std::cout << ptr - arr << std::endl; //print 3,
  //ptr - arr is the index of the element residing at ptr

  int arr2[] = {1, 3, 5}; //no even number
  int size2 = sizeof(arr2) / sizeof(arr2[0]);

  int* ptr2 = pointerToLastEven(arr2, size2);
  std::cout << ptr2 << std::endl; //nullptr is printed as 0x0

  return 0;
}

int* pointerToLastEven(int arr[], int size) {
  for (int index = size-1; index >= 0; index--) {
    if (arr[index] % 2 == 0) {
       return arr + index;
    }
  }

  return nullptr;
}
```

3 (10 points) Programming exercise on class

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].

Define non-member function `addInches`, given Length object len and integer parameter toAdd, the function should create and return a Length object that is the sum of `len` and `toAdd`. Example:

Suppose `len` is {2, 7} and `toAdd` is 30. Then the return of `addInches` function on the above parameters is {5, 1}.

Reason: 2 feet 7 inches is 2 * 12 + 7 = 31 inches. Add 31 by 30 is 61 inches, which equals 5 feet and 1 inch, where 5 is 61 divided by 12 and 1 is the remainder of 61 divided by 12.

Answer: The function is defined as follows.

```
Length addInches(Length len, int toAdd) {
  int totalInches = len.foot * 12 + len.inch + toAdd;

  return {totalInches / 12, totalInches % 12};
}
```

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

class Length {
```

```
public:
  int foot;
  int inch;
};

Length addInches(Length len, int toAdd);

int main() {
  Length len{2, 7};
  Length result = addInches(len, 30);

  std::cout << result.foot << " " << result.inch << std::endl; //5 1
  return 0;
}

Length addInches(Length len, int toAdd) {
  int totalInches = len.foot * 12 + len.inch + toAdd;

  return {totalInches / 12, totalInches % 12};
}
```

4 (10 points) Write codes of vector

Define a function called `shortStrings`, for a vector `v` of strings and an int `n`, search the `v`, find out the strings whose lengths are smaller than `n`, put them to a vector of strings, in the same order. Return that vector.

Given a vector of strings with elements `"hello", "", "hey", "hi"` and integer 3, the return is vector with elements `"", "hi"`.

Answer:

```
std::vector<std::string> shortStrings(std::vector<std::string> v, int n) {
  std::vector<std::string> result;
  for (int i = 0; i < v.size(); i++) {
    if (v[i].length() < n) {
      result.push_back(v[i]);
    }
  }

  return result;
}
```

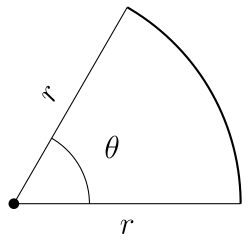
A complete code is shown as follows.

```
//Define a function called shortStrings, for a vector v of strings and an int n,
    search the v, find out the strings whose lengths are smaller than n, put them to a
     vector of strings, in the same order. Return that vector.
```

```

//Given a vector of strings with elements "hello", "", "hey", "hi" and integer 3, the
   return is vector with elements "", "hi".

#include <iostream>
#include <string>
#include <vector>

std::vector<std::string> shortStrings(std::vector<std::string> v, int n);

int main() {
  std::vector<std::string> vec = {"hello", "", "hey", "hi"};
  std::vector<std::string> result = shortStrings(vec, 3);

  for (int i = 0; i < result.size(); i++) {
    std::cout << "\"" << result[i] << "\"" << " ";
    //print "" "hi"
  }

  return 0;
}

std::vector<std::string> shortStrings(std::vector<std::string> v, int n) {
  std::vector<std::string> result;
  for (int i = 0; i < v.size(); i++) {
    if (v[i].length() < n) {
      result.push_back(v[i]);
    }
  }

  return result;
}
```

5 (15 points) Define class.

1. Define an Arc as a portion of a circle.



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

   (b) the **angle** between the two edges, represented as θ in the above figure, in **radians**

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

3. Define the default constructor, set data members **r** to be 1 and **angle** to be $\pi/4$, where π is represented by `M_PI` in C++.

   Answer:

```cpp
Arc::Arc() {
  r = 1;
  angle = M_PI / 4;
}
```

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

   (a) If r is positive and angle is strictly larger than 0 and angle is strictly smaller than 2π, where π is represented by `M_PI` in C++, set data member **r** by given parameter r and set data member **angle** by given parameter angle.

   (b) otherwise, set data members **r** to be 1 and **angle** to be $\pi/4$, where π is represented by `M_PI`.

   Answer:

```cpp
Arc::Arc(double r, double angle) {
  if (r > 0 && angle > 0 && angle < 2 * M_PI) {
    this->r = r;
    this->angle = angle;
  }
  else {
    this->r = 1;
    this->angle = 0.7854;
  }
}
```

5. Define method **getArea**, return the value of $\frac{1}{2}r^2$angle. Note that r and *angle* are data members.

   Answer:

```cpp
double Arc::getArea() const {
  return 1 / 2.0 * r * r * angle;
}
```

6. Define method **setRadius**, if given parameter r is positive, reset data member **r** by given parameter r.

   Answer:

```
void Arc::setRadius(double r) {
  if (r > 0) {
    this->r = r;
  }
}
```

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

7. Create an Arc object named **shape** from its non-default constructor with the radius of the arc as 2 and the angle as $\frac{\pi}{2}$, where π is defined as `M_PI` in C++.

   Answer:

```
  Arc shape(2, M_PI / 2);
```

8. Reset the radius to be 3.

   Answer:

```
  shape.setRadius(3);
```

9. Find out and print the area of shape.

   Answer:

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

Answer: A complete code is as follows.
code of Arc.hpp

```
#ifndef Arc_H
#define Arc_H
class Arc {
public:
  Arc();
  Arc(double r, double a);
  double getArea() const;
  double getPerimeter() const;
  void setRadius(double r);
  void setAngle(double angle);

private:
  double r; //radius of the arc
  double angle; //angle of the arc, represented in radians
};
#endif
```

Code of Arc.cpp

```cpp
#include "Arc.hpp"
#include <cmath>

Arc::Arc() {
  r = 1;
  angle = M_PI / 4;
}

Arc::Arc(double r, double angle) {
  if (r > 0 && angle > 0 && angle < 2 * M_PI) {
    this->r = r;
    this->angle = angle;
  }
  else {
    this->r = 1;
    this->angle = angle;
  }
}

void Arc::setRadius(double r) {
  if (r > 0) {
    this->r = r;
  }
}

void Arc::setAngle(double angle) {
  if (angle > 0 && angle < 2 * M_PI) {
    this->angle = angle;
  }
}

double Arc::getArea() const {
  return 1 / 2.0 * r * r * angle;
}

double Arc::getPerimeter() const {
  return 2 * r + r * angle;
}
```

code of TestArc.cpp

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

//sample output:
```

```
//area: 7.06858
//perimeter: 10.7124
int main() {
  Arc shape(2, M_PI / 2);
  shape.setRadius(3);
  std::cout << "area: " << shape.getArea() << std::endl;
  std::cout << "perimeter: " << shape.getPerimeter() << std::endl;

  return 0;
}
```

6 (10 points) function on vectors

Define a function called `mixNums`, given a vector of **int** type `vec`, do the following:

Check whether all the elements in `vec` has at least a postive number **and** at least a negative number.

For example, if `vec` has values 1, -2, 0, the returned is true since there is one positive number 1 and one negative number -2.

If `vec` has values 1, 2, 3, the returned is false since there is no negative numbers.

Answer: function `mixNums` is defined as follows.

```
bool mixNums(std::vector<int> vec) {
  int numPos = 0; //number of positive elements
  int numNeg = 0; //number of negative elements
  for (int i = 0; i < vec.size(); i++) {
    if (vec[i] > 0) {
      numPos++;
    }
    else if (vec[i] < 0) {
      numNeg++;
    }
  }

  return numPos > 0 && numNeg > 0;
}
```

A complete code is as follows.

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

bool mixNums(std::vector<int> vec);

int main() {
  std::vector<int> vec = {1, -2, 0};
  std::cout << std::boolalpha << mixNums(vec) << std::endl; //true

  std::vector<int> vec2 = {1, 2, 3, 4};
  std::cout << std::boolalpha << mixNums(vec2) << std::endl; //false

  return 0;
}

bool mixNums(std::vector<int> vec) {
  int numPos = 0; //number of positive elements
  int numNeg = 0; //number of negative elements
  for (int i = 0; i < vec.size(); i++) {
     if (vec[i] > 0) {
       numPos++;
     }
     else if (vec[i] < 0) {
       numNeg++;
     }
  }

  return numPos > 0 && numNeg > 0;
}
```

7 (10 points) Recursive Function

Define a recursive function `maxLen`, given an array of **string** with size many elements, return the maximum length of all strings.

For example, for array with elements `"hello", "hi", "how"`, the return is 5, since the longest string "hello" has 5 characters.

Hint: you should set the return type of `maxLen` as `size_t`, which is similar to `int` but represents only non-negative ingegers.

You may use `std::max` function from `algorithm` library, which takes two parameters of the same types and return the maximum. Or, you may use if-else statement to replace `std::max`.

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.

```
size_t maxLen(std::string arr[], int size) {
  if (size == 1) {
    return arr[0].length();
  }

  //maxLen(arr+1, size-1) is the maximum length of
  //the strings in the subarray of arr without arr[0].
  return std::max(arr[0].length(), maxLen(arr+1, size-1));
```

```cpp
  //Warning: std::min cannot compare an int and a size_t,
  //the types need to be the same.
  //length method of a string returns size_t type.

  //the above return statement can be rewritten as
//  size_t remMaxLen = maxLen(arr+1, size-1);
//  if (arr[0].length() > remMaxLen) {
//    return arr[0].length();
//  }
//  else {
//    return remMaxLen;
//  }
}
```

A complete code is as follows.

```cpp
//Purpose: define a recursive function to return the maximum lengths of strings in a
    given array.
#include <iostream>
#include <string>
#include <algorithm> //std::max

size_t maxLen(std::string arr[], int size);

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

  std::cout << maxLen(arr, size) << std::endl; //5,
  //the length of the longest string in arr is "hello",
  //which has 5 characters.
  return 0;
}

size_t maxLen(std::string arr[], int size) {
  if (size == 1) {
    return arr[0].length();
  }

  //maxLen(arr+1, size-1) is the maximum length of
  //the string in the subarray of arr without arr[0]
  return std::max(arr[0].length(), maxLen(arr+1, size-1));
  //Warning: std::min cannot compare an int and a size_t,
  //the types need to be the same.
  //length method of a string returns size_t type.

  //the above return statement can be rewritten as
//  size_t remMaxLen = maxLen(arr+1, size-1);
//  if (arr[0].length() > remMaxLen) {
//    return arr[0].length();
//  }
//  else {
//    return remMaxLen;
//  }
}
```