EECS 168 Final Exam Review Session

True/False and Multiple Choice

1. What kind of return statement is valid for a void function?
    a. return 0;
    b. return “”;
    c. return;
    d. No return statement is valid.

2. (T/F) All functions must have parameters.

3. The first element of an array is at index…
    a. -1
    b. 0
    c. 1
    d. 2

4. The last index of an array of size 5 is…
    a. 5
    b. -1
    c. 0
    d. 4

5. (T/F) Each parameter a function requires a type.

6. For loops are preferred when…
    a. The number of iterations is defined by a number or variable
    b. We want the loop to continue until a specific event happens
    c. The loop should be guaranteed to run at least once
    d. Never

7. (T/F) Passing an array to a function makes a copy of the entire array.

8. The size of a stack-allocated array must be known at…
    a. The beginning of the program
    b. Compile time (by the time we’re compiling the program)
    c. Runtime (by the time we’re running the program)
    d. It doesn’t matter

9. What is the correct syntax for creating a heap-allocated array of size “size”?
    a. int[] arr = new int* [size]
    b. int[] * arr = new int [size]
    c. int * arr = int [size]
    d. int * arr = new int [size]

10. (T/F) Passing a double (by value) to a function makes a copy of that double.

11. What is the correct syntax for creating a heap-allocated array (of size “size”) of pointers to “Triangle” objects?
    a. Triangle* arr = new Triangle*[size];

b. Triangle* arr = new Triangle[size];
c. Triangle** arr = new Triangle*[size];
d. Triangle* arr = new Triangle();

12. When would you want to create a heap-allocated array of pointers to objects?
    a. Never.
    b. When your class doesn't have a constructor with 0 parameters.
    c. When your class has a constructor with at least one parameter.
    d. When your class doesn't have a destructor.

13. Which of the following tells us we're allocating something in the heap rather than the stack?
    a. We are using pointers
    b. We are creating an array
    c. We are using the word "new"
    d. We are creating a 2D array

14. How do we get rid of stack-allocated arrays?
    a. We don't have to, the computer does it by itself
    b. Use the keyword delete[]
    c. Empty the array
    d. Set it to nullptr

15. What is the best way to avoid memory leaks?
    a. Not using pointers at all
    b. Using "delete" on everything created using "new"
    c. Setting our array pointer(s) to nullptr
    d. Using valgrind

Memory Management

Finish the definition for the main function such that there are no memory leaks. Assume Car is class with a single, default constructor (no parameters) and getters and setters for “year” and “make”. The Car class is defined in problem 4 of Code Writing. **Suggestion: Do Memory Allocation first (next page).**

```
int main()
{
        int size = 11;
        int nums[4];
        Car myCar;
        Car * myCar2 = new Car();

        Car arrayOfCars1[size];
        Car * arrayOfCars2 = new Car[size];

        Car ** arrayOfCarPointers = new Car*[size];
        for (int i = 0; i < size; i++)
        {
                arrayOfCarPointers[i] = new Car();
        }

        Car ** arrayOfCars2D = new Car*[size];
        for (int i = 0; i < size; i++) {
                arrayOfCars2D[i] = new Car[size];
        }

        bool temp = false;
        double myHeapAllocatedvariable = 1.5;

        //Your code here
```

Memory Allocation

Fill in the table regarding the variables in the Memory Management question (previous page).

Variable	Type	Stack or Heap?
size		
nums		
nums[1]		
myCar		
arrayOfCars1		
arrayOfCars1[0]		
myCar2		
arrayOfCars2		
arrayOfCars2[1]		
arrayOfCarPointers		
arrayOfCarPointers[3]		
arrayOfCars2D		
arrayOfCars2D[2]		
temp		
myHeapAllocatedVariable		

Code Tracing

Code that runs	Output to terminal

```
//Circle.h
class Circle {
	public:
		Circle(double radius) {
			m_radius = radius;
		}
		double calculateArea() {
			return 3.1416 * m_radius * m_radius;
		}
		void setRadius(double radius) {
			if (radius >= 0) {
				m_radius = radius;
			}
		}
		double getRadius() {
			return m_radius;
		}
	private:
		double m_radius;
};

//main.cpp
Circle* func2(int r) {
	Circle* circleObject = new Circle(r);
	return circleObject;
}

void func(char x, double& y, double& z, Circle* circlePtr) {
	x = 'n';
	circlePtr = func2(1);
	y = circlePtr->calculateArea();
	z = -2;
}

int main() {
	char x = 'm';
	double a = 4;
	double z = 3;
	Circle* ptr = nullptr;
	func(x, a, z, ptr);
	Circle* ptr2 = func2(4);
	cout << x << ", " << a << endl;

	if (ptr != nullptr) {
		 ptr->setRadius(z);
		cout << ptr->getRadius() << endl;
	}
	if (ptr2 != nullptr) {
		ptr2->setRadius(z);
		cout << ptr2->getRadius() << endl;
	}

	delete ptr;
	delete ptr2;
	return 0;
}
```

Code that runs	Output to terminal

```
char* reverse(char* arr, int& size) {
	size = 7;
	char* newArr = new char[size];
	for (int i = 0; i < size; i++) {
		newArr[i] = arr[size - 1 - i];
	}
	return newArr;
}

int main() {
	int size = 0;
	string myStr = "kwahyaj";
	char* myArr = new char[size];
	char* myOtherArr = nullptr;

	for (int i = 0; i < myStr.length(); i++)
	{
	  myArr[i] = myStr.at(myStr.length() - i - 1);
	}

	myOtherArr = reverse(myArr, size);

	for (int i = 0; i < size; i++)
	{
	  cout << myArr[i];
	}
	cout << '\n';
	cout << myStr << endl;

	char** my2DArr = new char*[size];
	for (int i = 0; i < size; i++)
	{
		my2DArr[i] = reverse(myOtherArr, size);
	}

	for (int i = 0; i < size; i++)
	{
		for (int j = 0; j < size; j++)
		{
			if (i >= j) {
				cout << my2DArr[i][j];
			}

		}
		cout << endl;
	}

	delete[] myArr;
	delete[] myOtherArr;
	for (int i = 0; i < size; i++)
	{
		delete[] my2DArr[i];
	}
	delete[] my2DArr;
	return 0;
}
```

Code Writing

Note: I added significantly more details to these questions since the original ones where a little vague.

1. Write a function that takes in a 2D array of integers and returns an array of the even integers in the 2D array. The function should take in the 2D array, the number of rows, the number of columns, and a variable for the size of the 1D array. The size of the 1D array can be initialized to 0 in main, but the function should modify the "size" variable in main to be equal to the amount of even integers.

2. Write a class StringClass that holds a heap-allocated array of characters.
   a. The function must have the following methods:
      - **StringClass(int length);** //Constructor that initializes the size and initializes the array
      - **~StringClass();**
      - **char getCharacter(int index) const;**
      - **void setCharacter(int index, char c);**
      - **int getLength() const;**

   b. Overload the assignment operator (write a function to use the assignment operator (the "=" operator) with StringClass objects and strings).

      bool operator==(std::string rhs) const;

      Example:
      StringClass myStrObject(8); //Pass in the size of the string
      myStrObject = "a string"; //In your function you can assume you'll get a string of the right size (in this case a string of length 8).

      Also, overload the comparison operator for equality (the "==" operator) so that we can verify whether two strings a StringClass and a string are equal.

      bool operator=(std::string rhs) const;

      Example:
      StringClass myStrObject(5);
      If (myStrObject == "Hello")
      {
      cout << "Equal!\n";
      }

   c. Write a copy constructor for a StringClass object. The copy constructor should copy the length, create a new array, and copy character by character.

      StringClass(const StringClass& strClassObject);

3. Write a class SocialMedia that holds an array of Users (User is a class). Obtain the number of users and each user's email, password, name, and age from a file called "data.txt".

Data.txt:
5
myEmail@socialmedia.com password Spongebob 20
emailAddresss@socialmedia.com 12345 Patrick 15
badEmail@socialmedia.com insecurePassword Squidward 40
UserEmail@socialmedia.com 54321 Gary 23
runningOutOfEmailNames@socialmedia.com abcde Plankton 32

The User class must contain a constructor with parameters for email, password, name, and age as well as getters and setters.

In the SocialMedia class make a "login" function that takes in an email and a password. The function should first verify if a user with that email exists. If such a user exists, the function should verify if that user's password matches the password passed in to the login function. Print a message telling the user whether the login was successful or not.

In the SocialMedia class also make a "search" function that takes in a char and prints all the Users whose name starts with that character and prints their name, age, and email address.

In main.cpp create a SocialMedia object and give the user options (login, search, or quit) until the user decides to quit.

4. Write code to print every Car object created in the program in the "Memory Allocation" question (including the Car objects that are inside arrays). Below are the Car.h and Car.cpp files.

```
//Car.h
#ifndef CAR_H
#define CAR_H

#include <string>

Class Car {
        private:
                string m_make;
                string m_model;
                int year;
        public:
                Car();
                void setMake(string make);
                string getMake() const;
                void setYear(int year);
                int getYear() const;
};

#endif
```

As stated in the "Memory Allocation" question, the Car constructor initializes make and year to "default" values. You won't necessarily use all the functions defined in this class. The goal of this problem is to use whatever functions you need to use to do all the printing.

```
//Car.cpp
#include “Car.h”
Car::Car()
{
	m_make = “Default Make”;
	m_year = 2018;
}
void Car::setMake(string make)
{
	m_make = make;
}
string Car::getMake() const
{
	return m_make;
}
void Car::setYear(int year)
{
	m_year = year;
}
Int Car::getYear() const
{
	return m_year;
}
```