EECS 168 2021 Spring Midterm: TR Version

Rules

- **DUE TIME:** You have until 11:59pm tonight to email your submission to your lab TA
- Email me all questions you have during the final
    - jwgibbo@ku.edu / jwgibbo@gmail.com
- DO NOT alter the formatting of the test. Any changes to formatting could result in grading errors
- DO NOT use wordpad (it alters the format)
- Only mark your answers within designated answer boxes
- Read and sign below
- Any strange characters you see are there on purpose
- **Unauthorized aid:** google searches, the materials or help of other students, past exams, help from the undergrad staff or GTAs, **any compilers** (e.g. don't just put the code problems in a compiler and run them), or **chegg.com**

I am fine with you using the following authorized aid:

- **Authorized aid**: your notes, your labs (the code, not the compiler), materials on the class website, your amazing brain

Your Name	Morgan Bergen
KUID	3073682

I'd like this exam to still be an assessment of your skill and understanding. If you agree to this, then please type the name of someone who would be heartbroken if they knew you cheated on this exam.

Their Name	My future employer, the controller of my funds and inital career existance.

[24pts] Conceptual

Provide your answers in the given boxes.

1. [2pts] When is a stack allocated variable that was declared in a function named *foo* deallocated?

   In general, the process of allocation and deallocation of stack memory are all executed automatically at the time of compilation. More specially speaking when the program runs, it will run in a specific order, after the function foo returns it's value and finishes it's execution, all the variables that are allocated within the stack memory from the function food gets deallocation.

2. [2pts] When is a heap allocated array that was created in a function named *foo* deallocated?

   As opposed to stack allocation, heap allocation is NOT done automatically, instead it is deallocated when a person deallocated it themselves at the end of the function. This is done specially by deleting the heap allocated array with the keyword delete followed by a pointer, which points to the heap allocated array, then thereafter the person must clear out the pointer by setting it equal to nullptr, these two lines of code are all written before the function gets returned.

3. [2pts] How many times, in a single object's life, is a constructor called?

   A constructor is called 1 time in the single objects life.

4. [2pts] How many times, in a single object's life, is a destructor called?

   A destructor is called 1 time in the single objects life.

5. [2pts] If an object is passed by value to a function, what special constructor is called?

   When an object is passed-by-value to a function a copy constructor is called.

6. [2pts] If a class has exactly one constructor and that constructor requires a parameter, can you make an array of objects of that type, yes or no?

Yes you can initialize an array of objects with that constructor.

7. [2pts] If we do not create our own copy constructor, does the default copy constructor create a deep or shallow copy?

The default copy constructor creates a shallow copy constructor.

8. [2pts] Why will the equation *area = (1/2)*base*height* always results in zero even if the base and height are greater than zero?

The reason as to why the equation will always result to zero is because the integers of (1/2) = 0. The integer 2 goes into 1 zero times and therefore anytime that is multiplied by 0 is also zero (regardless of its sign status as a positive number or a negative number).

9. [8pts, 2pts each] Assume a Circle class has a private member variable named m_radius and a public setRadius method. Assume "Circle.h" is included as needed. Answer the following questions below indicating whether or not the attempted access is legal or illegal by placing an 'X' in the appropriate column.

Code	Legal	Illegal

int main()
{
Circle c1;
c1.m_radius = 5.5.;
//main continues...
		x

int main()
{
Circle* c1 = new Circle();
c1->setRadius() = 5.5.;
//main continues...
		x

void changer(Circle& c)
{
c.setRadius(5.5); //written as intended
}
		x

void changer(Circle* c)
{
c.setRadius(5.5); //written as intended
}
		x

[22pts] Functions

For these questions you'll only need to write enough code to answer a given question.

[14pts] Define a function that takes an array of characters and its size. It returns true if the array contains the sequence 'd', 'o', 'g' (case-sensitive). The three letters MUST be in that exact order but can be anywhere in the array. You are NOT allowed to use any libraries/classes that perform the search for you. Assume valid parameters.
Examples:

Contents of array passed in	Return value
[a, b, c, d, o, g, x, y, z]	true
[g, o, d]	false
[d, o, a, g]	false
[d, o, g]	true
[d, o, g, d, o, g]	true

You will write your code in the box on the following page

```
#include <iostream>

bool dogSequence(char array[], int size);

int main(){

	//code to execute and call dogSequence function;

	return(0);
}

bool dogSequence(char array[], int size){
	int counter = 0;
	for (int i = 0, i < size; i++) {
		if ('d' == array[i]) {
			counter++;
			if ('o' == array[i + 1] {
				counter++;
				if ('g' == array[i + 2]) {
					counter++;
				}
			}
		}
	}
	if (counter >= 3) {
		return(true);
	} else {
		return(false);
	}
}
```

[8pts] Define a function that takes a 2D array of characters and its dimensions. This function will return true if the first and the last column are identical (case-sensitive). Assume valid parameters.

Examples:

Content of array passed in	return value

`a, z, q, a`
`b, t, p, b`
`y, r, d, y`
	true

`A, z, q, a`
`b, t, p, b`
`y, r, d, y`
	false

`a, a` `b, b`	true
`a,a,a,a,a` `a,a,a,a,z`	false

You will write your code in the box on the following page

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

bool identical(char **array, int rows, int cols){
	bool checked = false;
	std::string firstColumn = "";
	std::string lastColumn = "";
	for (int i = 0; i < rows; i++) {
		for (int j = 0; j < cols; j++) {
			if (j == 0) {
				firstColumn += array[i][j];
			} else if (j == cols - 1) {
				lastColunm += array[i][j];
			}
		}
	}
	if (firstColumn == lastColumn) {
		checked = true;
	}
	return (checked);
}

int main() {

	//code to execute and call identical function;

	return(0)
}
```

[26pts] Cool String: Part 1 - Memory

Below is the header file from our CoolString class. You may assume the methods listed here are working properly.

```
#ifndef COOL_STRING_H
#define COOL_STRING_H

class CoolString
{
      private:
      char* m_array;
      int m_size;

      public:
       //creates array of given size, stores size
      CoolString(int size);

       //makes deep copy
      CoolString(const CoolString& original);

      //delete array
      ~CoolString();

      //returns the size of the array
      int getSize() const;

      //return the character at an index
      char getEntry(int index) const;

       //stores character at an index
      bool setEntry(int index, char entry);

       // returns true is same size and all
       //value are in the same order
      bool operator==(const CoolString& rhs) const;

       //returns true if not the same (either differing size or
       //values)
      bool operator!=(const CoolString& rhs) const;
};
#endif
```

Using the CoolString header file for reference, carefully trace the following code then answer the questions below. You will trace the code up to the FREEZE POINT at which you may assume the program pauses (functions are paused in the middle of running and everything stays where it is in memory).

Code

```
void func(CoolString cs)
{
    cs.setEntry(0, 'H');
    //FREEZE POINT, func has NOT returned yet!

    return;
}

int main()
{

    std::string word = "village";
    CoolString myCS( word.length() );
    CoolString* cs168 = new CoolString(2);
    cs168->setEntry(0,'J');
    cs168->setEntry(1,'G');
    for(int i=0; i< myCS.size(); i++)
    {
        myCS.setEntry(i, word.at(i) );
    }

    myCS.setEntry(4, '8');//written as intended

    func(myCS);

    return(0);
}
```

1. [3pts] How many CoolString objects are allocated?

   2

2. [3pts] How many character arrays have been allocated by CoolStrings?

   1

3. [3pts] How many CoolString objects are on the call stack?

1

4. [3pts] How many CoolString objects are on the heap?

0

5. [3pts] How many character arrays are on the calls stack?

0

6. [3pts] How many character arrays are on the heap and being pointed to by CoolString member variables?

2

7. [4pts] For the instance named myCS, list the values within its array (e.g. if its array contained the characters C, A, T, S you would write those characters below).

vill8ge

8. [4pts] For the instance named cs, list the values within its array (e.g. if its array contained the characters C, A, T, S you would write those characters below).

Hillage

[28pts] Cool String: Part 2 - Implementation

1. [14pts] Assume you are adding a new method to the CoolString class called *isPalindrome()*. Assume its signature is the following in the header file:

```
#ifndef COOL_STRING_H
#define COOL_STRING_H
#include <iostream>
class CoolString
{
     //ASSUME ALL OTHER METHODS AND VARIABLES ARE STILL HERE BUT
     //OMITTED FOR SPACE SAKE
     bool isPalindrome() const;
};
#endif
```

The method isPalindrome returns true if the contents of the CoolString is a palindrome, meaning the characters in the array are the same forwards and backwards. It does NOT print anything. You may NOT alter the parameter list.

You will write your code in the box on the following page

```
//CoolString.cpp
#include <string>

bool CoolString::isPalindrome() const
{
  //Your code below

      bool checker = false;
      int SIZE = int(strlen(m_array));
      char storage[size];
      char reverse[size];
      std::string original = "";
      std::string reversed = "";

      for (int i = 0; i < SIZE; i++){
            storage[i] = m_array[i];
            original = original = storage[i];
      }
      for (int i = 0; i < SIZE; i++){
            reverse[i] = stroage[SIZE - i - 1];
            reversed = reversed + reverse[i];
      }
      if (original == reversed) {
            checker = true;
      } else {
            checker = false;
      }
      return(checker);
}
```

2. [14pts] Assume you are in main.cpp, write a function definition for a function named *fillFromFile* that takes a string that represents a file name. The file will contain an integer (e.g. 5) then a series of characters, all values on their own line. The number in the file represents how many characters will follow (e.g. 3 then C A T might be in a given file). Using the contents in the file, create a heap-allocated CoolString of the appropriate size and fill it with the characters from the file. Return a pointer to the CoolString object.

```
//main.cpp
CoolString* fillFromFile(std::string fileName)
{
   //your definition below
     int SIZE = 0;
     CoolString* heapString = nullptr;

     std::ifsteam inFile;
     inFile.open(fileName);
     if (inFile.is_openn()) {

     inFile >> SIZE;
     for (int i = 0; i < SIZE; i++){
          inFile >> heapString[i];
     }
     inFile.close();
     } else {
          std::cerr << "File could not be opened. \n";
     }

     return(*heapString);
}
```

This should be page 15.

Submit your exam to your lab TA via email 11:59pm CST