# EECS 168 2021 Summer Final

# 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](mailto:jwgibbo@ku.edu) / [jwgibbo@gmail.com](mailto: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
* **Unauthorized aid:** google searches, the materials or help of other students, past exams, help from the undergrad staff or GTAs, or **chegg.com**
* **Authorized aid**: your notes, your labs (the code, not the compiler), materials on the class website, your amazing brain

|  |  |
| --- | --- |
| Your Name | Matthew McManness |
| KUID | 2210261 |

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 | Brenda McManness |

# [1pt] Honesty agreement

I'd like this exam to still be an assessment of your skill and understanding, and not your ability to find answers online. Please type the name of someone who would be heartbroken if they knew you cheated on this exam.

|  |  |
| --- | --- |
| Their Name | Brenda McManness |

# 

# [13pts] 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?

|  |
| --- |
| When the function (foo) ends (returns). |

1. [2pts] When is a heap allocated array that was created in a function named *foo* deallocated?

|  |
| --- |
| Only when it is deleted from the heap ( delete[ ] ). |

1. [2pts] How many times, in a single object’s life, is a constructor called?

|  |
| --- |
| Once and only once. |

1. [2pts] How many times, in a single object’s life, is a destructor called?

|  |
| --- |
| Once and only once. |

1. [2pts] If an object is passed by value to a function, what special constructor is called?

|  |
| --- |
| A copy constructor. |

1. [2pts] What kind of method cannot take parameters?

|  |
| --- |
| Destructor methods do not take parameters. |

1. [1pts] If we do not create our own copy constructor, does the default copy constructor create a deep or shallow copy?

|  |
| --- |
| Shallow copy |

# [40pts] Code writing

1. [10pts] Write a function definition for a function named *arrayToFile*. This function takes an array of doubles, its size, and a std::string representing a filename. Your function will open and write all elements of the array to the file with each element on its own line.

**Write your code on the space provided on the following page**

|  |
| --- |
| #include <fstream>  #include <string>  using namespace std;    Void arryToFile(double\* array, int size, string fileName)  {  ofstream myOutputFile;  myOutputFile.open(fileName);    for(int i = 0; i < size; i++)  {  myOutputFile << array[i] << '\n';  }    myOutputFile.close();  } |

1. [7pts] Assume there is a function named *dangerous(double num)* that is already defined but you aren't allowed to see its definition. All you know is that you pass it a double and it can either return a double or throw a std::runtime\_error. The conditions that cause it to throw an exception are unknown to you. You must write a program that will let the user pick the number you pass to dangerous. You must prompt the user over and over until you get a value out of dangerous. If dangerous throws an exception, simply ask the user for another number and try again.

**Write your code on the space provided on the following page**

|  |
| --- |
| //  I have created two versions and both will work:  #include <iostream>  #include <stdexcept>  using namespace std;    int main()  {  double num = 0;  bool errorCatch = 1;    while(errorCatch = 1)  {  try  {  cout << "choose a double to run through the dangerous function: ";  cin >> num;  dangerous(num);  errorCatch = 0;  }  catch(runtime\_error& rte)  {  cout << "Something went wrong.\n\n";  }  }  return(0);  }    #include <iostream>  #include <stdexcept>  using namespace std;    int main()  {  double num = 0;    while(true)  {  try  {  cout << "choose a double to run through the dangerous function: ";  cin >> num;  dangerous(num);  break;  }  catch(runtime\_error& rte)  {  cout << "Something went wrong.\n\n";  }  }  return(0);  } |

1. [5pts] Write a function definition for a function named *aboveAverage*. This function takes an array of doubles, its size and returns a count of how many values in the array are above the average value in the array. For example if an array has 1000 values and the average value is 55.5 you will need to return a count of how many of the 1000 values are strictly above 55.5.

**Write your code on the space provided on the following page**

|  |
| --- |
| int aboveAverage(double\* array, int size)  {  int aboveAverageCount = 0;  double average = 0.0;    for(int i = 0; i < size; i++)  {  average = average + array[i];  }  average = average/size;    for(int i = 0; i < size; i++)  {  if(array[i] > average)  {  aboveAverageCount++;  }  }  return(aboveAverageCount);  } |

1. [12pts] Write a function definition for a function named *mirror2D.* This function takes a 2D array of characters and its dimensions. It will modify the array so that after your function ends, the array's values are "mirrored". See example:

|  |  |
| --- | --- |
| **Original array** | **Array after call** |
| [A, B, C, D]  [E, F, G, H]  [I, J, K, L] | [D, C, B, A]  [H, G, F, E]  [L, K, J, I] |
| [T, A, C, O, S]  [R, A, C, O, S]  [B, A, C, O, S]  [M, A, C, O, S] | [S, O, C, A, T]  [S, O, C, A, R]  [S, O, C, A, B]  [S, O, C, A, M] |

**Write your code on the space provided on the following page**

|  |
| --- |
| void mirror2D(Char\*\* array2D, int rows, int cols)  {  for(int i = 0; i < rows; i++)  {  for(int j = 1; j <= cols; j++)  {  array2D[i][(j-1)] = array2D[i][(cols-j)];  }  }  return(array2D);  } |

1. [6pts] Write a complete main.cpp that will take in two command line arguments that represent words. Your program will print the longer of the two words or print "SAME" if they are the same.

**Write your code on the space provided on the following page**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;    int main(int argc, char\* argv[])  {  string word1 = "";  string word2 = "";    if(argc == 3)  {  word1 = argv[1];  word2 = argv[2];  if (word1.length() > word2.length())  {  cout << word1 << '\n';  }  else if (word2.length() > word1.length())  {  cout << word2 << '\n';  }  else  {  cout << "SAME\n";  }    }  else  {  std::cout << "Wrong number of command line arguments!\n";  }  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 length 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(4, 'R');  //FREEZE POINT, func has NOT returned yet!  //Now go answer the questions below  return;  }  int main()  {  std::string word = "XOR3BOO";  CoolString myCS( word.length() );  for(int i=0; i< myCS.getSize(); i++)  {  myCS.setEntry(i, word.at(i) );  }  myCS.setEntry(0, 'S');  func(myCS);    return(0);  } |

1. [3pts] How many CoolString objects (not including std::string) are allocated?

|  |
| --- |
| 2 |

1. [3pts] How many character arrays are allocated by CoolStrings?

|  |
| --- |
| 1 |

1. [3pts] How many CoolString objects are on the call stack?

|  |
| --- |
| 2 |

1. [3pts] How many CoolString objects are on the heap?

|  |
| --- |
| 0 |

1. [3pts] How many character arrays controlled by CoolStrings are on the call stack?

|  |
| --- |
| 0 |

1. [3pts] How many character arrays controlled by CoolStrings are on the heap?

|  |
| --- |
| 1 |

1. [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).

|  |
| --- |
| SOR3BOO |

1. [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).

|  |
| --- |
| SOR3BOO |

# [20pts] Cool String: Part 2 - Implementation

1. [10pts] Assume you are adding a new method to the CoolString class called replace. 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  void replace(char target, char exchange);  };  #endif |

The method *replace* will take a target character and an exchange character. It will replace all occurrences of the target character with the exchange character. For example, if the CoolString's array contains [A, B, C, D, B, A] and the target is 'B' and the exchange is '?' then after *replace* runs, the array will contain [A, ?, C, D, ?, A]. You may NOT alter the parameter list.

**Write your code on the space provided on the following page**

|  |
| --- |
| //CoolString.cpp    void CoolString::replace(char target, char exchange)    {      //Your code below    for(int i = 0; i < size\_m; i++)  {  if(target == this->getEntry(i))  {  this->setEntry(i, exchange);  }  }  } |

1. [10pts] 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.

**Write your code on the space provided on the following page**

|  |
| --- |
| //main.cpp  CoolString\* fillFromFile(std::string fileName)  {  std::ifstream myInputFile;  myInputFile.open(fileName);  int size = 0;  char target = '\0';  myInputFile >> size;  CoolString\* myCS = nullptr;  myCS = new CoolString(size);    for(int i = 0; i < size; i++)  {  myInputFile >> target;  myCS.setEntry(i, target);  }  return(myCS)  } |

**This should be page 22.**

**Submit your exam to your TA via email**