CSE278 SYSTEMS 1
Department of Computer Science and Software Engineering
Miami University
Spring 2020
Mid Term Exam
March 12, 2020
Full Marks: 100
There are 15 Questions for a total of 100 points.
Answer ALL the questions

NAME: **William Mechler** **MUID: mechlewg**

General Instructions

- First, ensure you have all the 10 pages of this exam booklet even before starting
- This exam is closed notes and closed books. No discussions are permitted
- You may use Calculator
- Do not bring out your cell phone; don't answer the phone; don't read text messages
- You have 2 hours to complete the exam
- Write your answers clearly
- The size of the space given for each answer is sufficient
- Write no more than 3-4 lines for each of the short questions
- Show all your works for the Mathematical problems
- Even if your final answers are incorrect, you will get partial credit if intermediate steps are clearly shown to highlight thought process. This applies to program tracing questions as well.

GOOD LUCK!

SHORT QUESTIONS.

[5 * 10 = 50 point]

1. Briefly (1 to 2 sentences each) state the 4 key principles of object-oriented programming

The 4 key principles of object-oriented programming are encapsulation, abstraction, inheritance and polymorphism.

2. Elaborate the meaning of the following as well as mention the port number that they use:

Protocol	**Elaborated meaning**	**Port Number**
SMTP	Simple Mail Transfer Protocol	25
HTTP	Hypertext Transfer Protocol	80
POP3	Post Office Protocol	110
IMAP	Internet Message Access Protocol	143
HTTPS	Hypertext Transfer Protocol over TLS/SSL	443

3. For a communication session between a pair of processes, which process is the client and which is the server?

The process which initiates the communication is the client; the process that waits to be contacted is the server.

4. What information is used by a process running on one host to identify a process running on another host?

The IP address of the destination host and port number of the destination socket.

5. What is the difference between unordered_map and vector?

A vector is an arraylist of elements of the same data type where you access through their index. While a unordered_map is a collection of key and value pairs where you access the information by their key.

6. What do you understand by the term CSV? Consider the following contents of a file, faculty.txt:

   054, Campbell
   103, Kiper
   112, Rao

   a. What UNIX/Linux command can be used to produce a output, list just the faculty names:

   Campbell
   Kiper
   Rao

```
awk 'BEGIN  { FS = "," } ; { print $1 }'
```

   b. What C++ function/code that can be used to produce the similar output for the above. Make necessary assumptions.

```
while(std::getline(myFile,line)){
  std::stringstream ss(line);
  int col = 0;
```

```
  while(ss > val){
    result.at(col).second.push_back(val);
  if(ss.peek() == ',') ss.ignore();
  col++;
  }
}
```

7. How is a well-known port different from an ephemeral port?

Well-known ports are numbered below 1023 when Ephemeral ports are numbered above 1024.

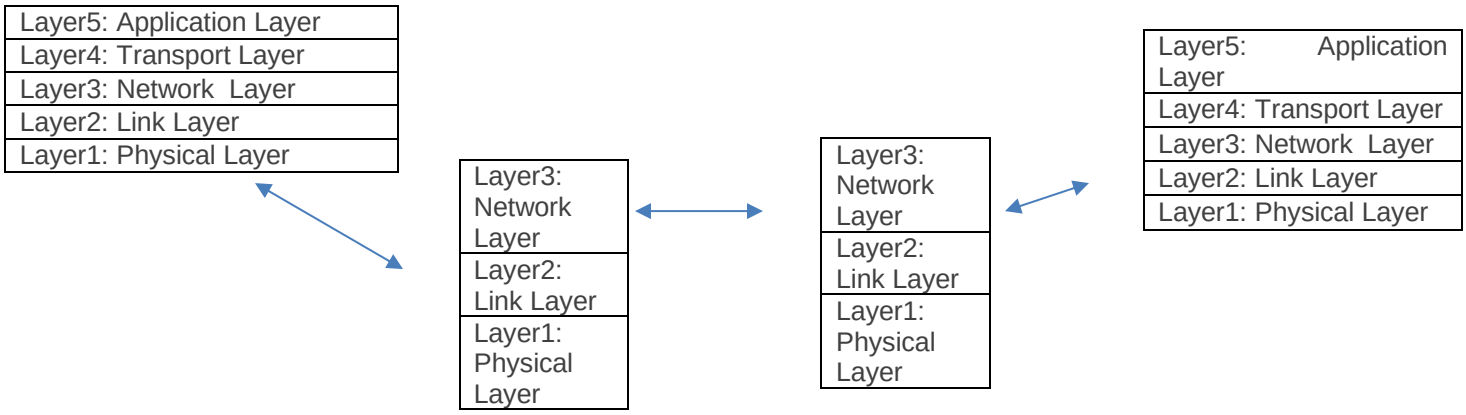
8. A user on the host 170.210.17.145 is using a Web browser to visit www.example.com. In order to resolve the www.example.com domain to an IP address, a query is sent to an authoritative name server for the "example.com" domain. In response, the name server returns a list of four IP addresses in the following order {192.168.0.1, 128.0.0.1, 200.47.57.1, 170.210.17.130}. Even though it is the last address in the list returned by the name server, the Web browser creates a connection to 170.210.17.130. Why?

It connections to 170.210.17.130 because it is the most simple and closest address so it is the fastest connection possible.

9. A Host has an IP address of 10001000 11100101 11001001 00010000. Find the *class* and *decimal equivalence* of the IP address.

Decimal equivalence of IP address = 136.229.201.16
IP address class = Class 2

10. Draw a TCP/IP Internet that consists of two networks connected by a router. Show a computer attached to each network. Show the protocol stack used on the computers and the stack used on the router



C++ Coding Problems

11. [10] Develop a simple C++ program that:

a. Reads words (separated by whitespace only and not any other punctuation or special characters) from the user until logical end-of-file

b. Prints the just the count of words (just one integer and no other text/messages) that start with an English vowel, i.e., one of: AEIOUaeiou characters.

For example, given just the sentence “Elephants are Awesome animals” the output will be 4. Similarly, for the input “I think I am warming up to C++” the output is also 4.

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

bool isVowel(char test);

Int main(){

 String word “”;
  Int Vowel = 0;
          While(cin >> word)
          {
```

```
            If(isVowel(word[0]){
              Vowel++;
             }

           }

           cout << Vowel;
}

bool isVowel(char test){

  if(test = 'a' || test = 'A' || test = 'e' || test = 'E' || test = 'i' || test = 'I'){
    if(test ='o' || test ='O' || test = 'u' || test = 'U'){

        return true;
     }

   }
  return false;
}
```

12. [5] Assume that a system has a 32-bit virtual address with a 4-KB (***1 KB = 1024***) page size. Write a C++ program that is passed a virtual address (in decimal) on the command line and have it output the page number and offset for the given address. As an example, your program would run as follows:

./a.out 19986

Your program would output:

The address 19986 contains:

page number = 4

offset = 3602

```
Int main(){

 unsigned int address = stoi(argv[1]);
 int pageCount = (address / 4096);
 int offset = (address % 4096);
 cout << "The address << address << contains: << "\n";
 cout << "page number = " << pageCount << "\n";
 cout << "offset = " <<    << "\n";

}
```

13. [10] Complete the following method that returns a vector with only *odd* values in the src vector. If the src vector has values {2, -4, 7, 9, 3, 8} this method should return a vector with values {7, 9, 3}.

```
using IntVec = std::vector<int>;

IntVec odds(const IntVec& src) {

Vector<int> oddNumbers;

for(int i = 0; i < src.size(); i++){

        if(src[i] % 2 != 0){

        oddNumbers.push_back(src[i]);

        }

   }

Return oddNumbers;

}
```

14. [10] File I/O using file streams

Develop a C++ program that prints the *first* line of each paragraph in a given text file specified as command-line argument. Paragraphs are separated by one or more blank (i.e., empty) lines.

a. In order gain practice working with generic I/O streams, add the following method that works with abstract streams to your starter code:

void printFirstLine(std::istream& is, std::ostream& os)

The above method is the one that should process lines from input stream is (use std::getline to read lines) and print *first* line of each paragraph to output stream os.

b. Call the printFirstLine method (from main) with a suitable std::ifstream to process data from a given text file specified as a command-line argument. Recollect that the file name will be in argv[1] in the main method. Use std::cout as the output stream.

```
#include <cstdlib>

#include <fstream>

#include <iostream>

#include <string>

#include <algorithm>

#include <iterator>
```

```
#include <sstream>
#include <fstream>
#include <vector>
using namespace std;

void printAllLine(std::istream& is, std::ostream& os){
    std::string line;

    while (std::getline(is, line)) {
        std::cout << line << std::endl;
    }
}
void printLastLine(std::istream& is, std::ostream& os) {
    std::string line, prevLine;

    while (std::getline(is, line)) {
        if (line.empty() && !prevLine.empty())
            //std::cout << prevLine << std::endl;
            os << prevLine << std::endl;
        prevLine = line;
    }
    if (!prevLine.empty()) {
        //std::cout << prevLine << std::endl;
        os << prevLine << std::endl;
    }
}
std::vector<std::string> split (const std::string& line){
    std::vector<std::string> words;
```

```
    std::istringstream is(line);
    std::string word;
    const char delimiter = ' ';
    while (std::getline(is, word, delimiter)){
        words.push_back(word);
    }
    return words;
}
int main(int argc, char** argv) {
    std::vector<std::string> myVector;
    std:string myLine = "This is a test line to split";
    std::ifstream paras(argv[1]);
    std::ofstream parasOut(argv[2]);
    //printAllLine(paras, std::cout);
    printLastLine(paras, parasOut);
    myVector = split(myLine);
    for (auto word: myVector){
        std::cout << word << std::endl;
    }
    return 0;
}
```

```
void printFristLine(std::istream& is, std::ostream& os){
    std::string line, prevLine;

    while (std::getline(is, line)) {
        if (!line.empty() && prevLine.empty())
            std::cout << prevLine << std::endl;
            os << prevLine << std::endl;
        prevLine = line;
    }
    if (!prevLine.empty()) {
        //std::cout << prevLine << std::endl;
        os << prevLine << std::endl;
    }
```

```
}
```

15. [15] Consider the following GradeBook class definition and implementation:

```
// Fig. 3.11: GradeBook.h GradeBook class definition. This file presents GradeBook's public
//interface without revealing the implementations of GradeBook's member
// functions, which are defined in GradeBook.cpp.
#include <string> // class GradeBook uses C++ standard string class
using std::string;

// GradeBook class definition
class GradeBook
{
public:
   GradeBook( string, string ); // constructor that initializes courseName and instructorName
   void setCourseName( string ); // function that sets the course name
   string getCourseName(); // function that gets the course name
   void setInstructor( string ); // function that sets the instructor's name
   string getInstructor();      // function that gets the instructor's name
   void displayMessage(); // function that displays a welcome message
private:
   string courseName; // course name for this GradeBook
   string instructor; // course's instructor's name
      }; // end class GradeBook

// Fig. 3.12: GradeBook.cpp
// GradeBook member-function definitions. This file contains
```

```
// implementations of the member functions prototyped in GradeBook.h.
#include <iostream>
using std::cout;
using std::endl;
#include "GradeBook.h" // include definition of class GradeBook
// constructor initializes courseName with string supplied as argument
GradeBook::GradeBook( string name, string instructor )
{
   setCourseName( name ); // call set function to initialize courseName
   set setInstructor ( instructor ); // call set function to initialize instructorName
} // end GradeBook constructor
// function to set the course name
void GradeBook::setCourseName( string name )
{
   courseName = name; // store the course name in the object
} // end function setCourseName

// function to get the course name
string GradeBook::getCourseName()
{
   return courseName; // return object's courseName
} // end function getCourseName

void GradeBook::setInstructor ( string name )
{
   instructor = name; // store the instructor name in the object
} // end function setInstructor
```

```
string GradeBook::getInstructor ()

{

   return instructor; // return object's instructor

} // end function getInstructor
```

```
// display a welcome message to the GradeBook user

void GradeBook::displayMessage()

{

   // call getCourseName to get the courseName

   cout << "Welcome to the grade book for\n" << getCourseName() << "This course is presented by\n" <<
getInstructor();

      << "!" << endl;

      } // end function displayMessage
```

(*Modifying Class GradeBook*) Modify class GradeBook as follows:

- **a.** Include a second string data member that represents the course instructor's name.
- **b.** Provide a *set* function to change the instructor's name and a *get* function to retrieve it.
- **c.** Modify the constructor to specify two parameters – one for the course name and one for the instructor's name.
- **d.** Modify member function displayMessage such that it first outputs the welcome message and course name, then outputs "This course is presented by:" followed by the instructor's name.

Use your modified class in a test program that demonstrates the class's new capabilities.

```
#include <iostream>

int main() {
  GradeBook test GradeBook("CSE278", "Emded");
  cout << test.getCourseName() << test.getInstructor();
  test.setCourseName("MTH151");
  test.setInstructor("Mark");
  cout << test.getCourseName() << test.getInstructor();
}
```