5.1 Description

The first part of this assignment will be a brief design proposal for your assignment 6. Assignment 6 you will get to build and explore whatever peaks your interest. This could be implementing a data structure or algorithm, building several classes that can be re-used, exploring computer graphics, databases, or otherwise integrating external tools with C++.

The second part of this assignment is putting to use our Object-Oriented knowledge. We are going to be building a class for performing matrix math operations (You can think of this assignment as an update to the Analytical Engine—we can do more powerful things). You will be given the interface file, and then it is up to you to make sure your implementation matches the sample solution.

\[
\begin{bmatrix}
  a & b & c \\
  d & e & f \\
  g & h & i
\end{bmatrix}
\times
\begin{bmatrix}
  j & k & l \\
  m & n & o \\
  p & q & r
\end{bmatrix}
= \\
\begin{bmatrix}
  aj + bm + cp & ak + bn + cq & al + bo + cr \\
  dj + em + fp & dk + en + fq & dl + eo + fr \\
  gj + hm + ip & gk + hn + iq & gl + ho + ir
\end{bmatrix}
\]
5.2 Task

The tasks for this assignment are the following:

1. Write your design proposal. This will include the following:
   (a) Name of the project (give it a descriptive and cool title)
   (b) A 3-5 sentence description of what it will do.
   (c) Draw or make one diagram of the classes and diagrams you think you will need in this program—show how they interlink. You can draw it on paper and take a picture, use PowerPoint, Photoshop, or any other tools you need to make the diagram.
   (d) Provide a bulleted list of features to make the minimum viable program (MVP) (i.e. What you consider would earn you an A on this assignment).
   (e) A bulleted list of the maximum deliverable (All of the features you would want in your program, perhaps if you had a full semester to work on it).

2. You will be given the interface to implement a matrix math library. It is your job to fill in the program logic.
   - Implement each of the functions in a separate matrix.cpp file.
   - Make sure that the functions are properly templated.
   - Ensure there are no memory leaks or segmentation faults.
   - Make sure the math works!

5.3 Input Files
```cpp
#pragma once

#include <iostream>
#include <string>

// CompMatrix
// Purpose: A 4x4 Matrix Library of functions useful
//           for math and graphics applications
// Notes: You may add whatever functionality you like or
//        any additional member functions that make implementation
//        easier.
template <class T>
class CompMatrix{
  T** m_elements;

  // allocate (Optional)
  // Desc: Helper function to allocate member in all of our
  //        constructors consistently.
  void allocate();

  public:
    // Constructor
    // Desc: The constructor creates a matrix filled with 0.
    //       This is a multi-dimensional matrix.
    CompMatrix();

    // Destructor
    // Desc: Frees memory
    //       Note that m_elements consists of arrays with allocated
    //       memory
    ~CompMatrix();

    // Copy Constructor
    // Desc:
    CompMatrix(const CompMatrix<T> &rhs);

    // = Assignment operator
    // Desc: Assignment operator which performs a copy.
    //       Sets each element of a matrix equal to the other
    CompMatrix& operator =(CompMatrix const &rhs);

    // reset
    // Desc: Sets all of the value in the matrix to 0
    void zero();

    // identity
    // Desc: Sets the current matrix to the identity matrix
    //       The diagonal(top-left to bottom-right) are 1’s
    //       and everything else is a 0
    void identity();

    // * Matrix Multiply
    // Desc: This function multiplies a matrix by another one
    // Tutorial on Matrix Multiplication:
    // https://www.khanacademy.org/math/precalculus/precalc-matrices/
    // multiplying-matrices-by-matrices/v/multiplying-a-matrix-by-a-
}```
matrix
CompMatrix operator *(CompMatrix const &m);

// + (Add operator)
// Desc: This operator (function) allows for matrix addition
CompMatrix operator +(CompMatrix const &m);

// - (Subtract operator)
// Desc: This operator (function) allows for matrix subtraction
CompMatrix operator -(CompMatrix const &m);

// set
// Desc: Sets the value of a matrix at a specified row and column.
void set(unsigned int row, unsigned int column, T value);

// at
// Desc: Returns the value at a specified row and column.
// Note that the const at the end means no member variables are
// modified within this function when T is returned.
T at(unsigned int row, unsigned int column) const;

// print
// Desc: prints out the current matrix to the terminal with a
// message above it
void print(std::string msg);

// (Optional) - You may choose to implement [] if you find it
// convenient
// [] (Access operator)
// CompMatrix operator [](CompMatrix const &m);
}
#endif

Listing 5.1: matrix.h interface
5.4 Files

You will be creating the files from scratch for the project proposal. For the matrix math library, a matrix.h file is provided.

5-4
You will have files named exactly (including the exact capitalization):

- proposal.pdf or proposal.docx
- main.cpp matrix.cpp matrix.h
- README

## 5.5 Submission

You must test your program on the machines in Halligan (either physically or by ssh'ing into the homework.cs.tufts.edu server).

```bash
provide comp11 ps5 main.cpp matrix.cpp matrix.h README proposal.pdf
```

Listing 5.4: Submit Assignment

## 5.6 Expected Output

Your provided code samples should compile and run on the Halligan Machines. There should not be memory leaks!

A reference solution is provided here (log into the halligan machines and run):

`/comp/11/solution/ps5`

## 5.7 Evaluation/Rubric

- Make sure you test your program on the halligan computers (either physically or through ssh).
- Files are properly named as specified below.
- You include a file called README describing your assignment.
- Part of your assignment will be graded based on code style.
- The remaining part of your assignment will be graded based on the correctness, and that it works to the specification.
  - You project proposal should include all of the items in the tasks at a minimum (You may add more if you like).
  - Your proposal should be spell checked.
  - Your proposal should be concise, but also complete enough that I handed it to another student they would understand what they need to do to implement the assignment.
5.7.1 Style

Use good style, and liberally use comments! Some examples are provided below that you may use as a template.

```cpp
// = = = = = = = = = = = = = = = = = = = = = = = = = = if−statements = = = = = = = = = = = = = = = = = = = =
if (a > b) {
  if (b > c) {
    // Notice how the indentation nicely lines up with the conditionals
  }
}
// = = = = = = = = = = = = = = = = = = = = = = = = = = for−loops = = = = = = = = = = = = = = = = = = = =
std::string s = "hello";
for (char c : s) {
  // Range-based for loops are useful if we are not modifying the value.
  std::cout << c;
}
int myArray[10];
for (int i = 0; i < 10; ++i) {
  // If we are modifying values, then use a regular for−loop
  myArray[i] = i;
}
// = = = = = = = = = = = = = = = = = = = = = = = = = = structs = = = = = = = = = = = = = = = = = = = =
// Typically, a struct is known as a POD data structure.
// structs typically do not have member functions (although they are allowed to).
// Instead, we use classes as a convention.
// However, we will write one of the 3 amigos (Constructor, Destructor, Copy Constructor) if we are allocating memory.

struct node {
  node* node;
  int data;
};
// = = = = = = = = = = = = = = = = = = = = = = = = = = classes = = = = = = = = = = = = = = = = = = = =
// Purpose: This class creates objects that perform math operations
// Example Usage:
// myClass c;
// c.print();

class myClass {
public:
  // Desc: This returns a 1 if the integer value is even. Otherwise 0 is returned.
  // Example: Returns a 1 or 0 integer
  // Error Conditions: User inputs a non−integer value
  int isEven();
  private:
};
```

Listing 5.5: Some Good Style Examples