CSC 216 Portfolio 1

Nicolas Nytko

December 7, 2016

Contents

1 Homework

1.1 Linked Lists

1.1.1 Problem R-3.7

> *Give an algorithm for finding the penultimate (second to last) node in a singly linked list where the last element is indicated by a null next link.*

Listing 1: ../hw/r-3.7.cpp

```cpp
node* getPenultimateNode( node* pStart )
{
    // Check to see if our starting node is NULL or not

    if ( pStart != NULL )
    {
        // Make sure we have more than 1 node

        if ( pStart->next == NULL )
            return pStart;

        node* pCurrent = pStart;

        while ( pCurrent->next->next != NULL ||
               ( pCurrent->next->next == NULL && pCurrent->next != NULL ) )
        {
            pCurrent = pCurrent->next;
        }

        return pCurrent;
    }
    else
    {
        return NULL;
    }
}
```

1.1.2 Problem R-3.10

> *Describe a nonrecursive function for finding, by link hopping, the middle node of a doubly linked list with header and trailer sentinels. (Note: This function must only use link hopping; it cannot use a counter.) What is the running time of this function?*

Listing 2: ../hw/r-3.10.cpp

```cpp
node* getMiddleNode( list* pList )
{
    if ( pList == NULL )
        return NULL;

    // Have two nodes, one that starts at the beginning of the list and one that
      starts at the end

    node* pStart, pEnd;
    pStart = list->start;
    pEnd = list->end;

    // Loop until the nodes "intersect" and point to the same data

    while ( !( pStart == pEnd || ( pStart->next == pEnd && pEnd->prev == pStart ) )
        )
    {
        pStart = pStart->next;
        pEnd = pEnd->prev;
    }

    return pStart;
}
```

1.1.3 Problem C-3.3

> *Let A be an array of size $n \geq 2$ containing integers from 1 to $n-1$, inclusive, with exactly one repeated. Describe a fast algorithm for finding the integer in A that is repeated.*

Listing 3: ../hw/c-3.3.cpp

```
int getRepeatNumber( int* pArray, size_t nLength )
{
    bool bFound = false;
    int nRepeat = -1;

    for ( size_t i=0; i < nLength && !bFound; i++ )
    {
        for ( size_t j=i+1; j < nLength && !bFound; j++ )
        {
            if ( pArray[i] == pArray[j] )
            {
                nRepeat = pArray[i];
                bFound = true;
            }
        }
    }

    return nRepeat;
}
```

1.1.4 Problem C-3.4

> *Let B be an array of size $n \geq 6$ containing integers from 1 to $n-5$, inclusive, with exactly five repeated. Describe a good algorithm for finding the five integers in B that are repeated.*

Sort B. Create a variable that holds how many times the current value has been repeated, and a variable to hold what the last value was. Loop through the array and increment the repeat variable every time the value has been repeated, and reset it to 0 when a new value is introduced. End when the repeat value reaches 5.

1.1.5 Problem C-3.5

> *Suppose you are designing a multi-player game that has $n \geq 1000$ players, numbered 1 to n, interacting in an enchanted forest. The winner of this game is the first player who can meet all the other players at least once (ties are allowed Assuming that there is a function*

meet(i, j), which is called each time a player i meets a player j (with $i \neq j$), describe a way to keep track of the pairs of meeting players and who is the winner.

Create a two dimensional boolean array with the x axis corresponding to the index of the player, and the y axis corresponding to whether that person was met. So to see if player x has met player y, check array index `p[x][y]`. If a full column is filled, then that player has met everyone.

1.1.6 Problem C-3.8

Describe a good algorithm for concatenating two singly linked lists L and M, with header sentinels, into a single list L' that contains all the nodes of L followed by all the nodes of M. Create a new list L' and copy all of the nodes from L into it. At the end of L', copy all of the nodes of M.

Copy all of the elements from L into L'. Start copying M at the last node of L' after the header sentinel and continue until the null pointer.

1.1.7 Problem C-3.9

Give a fast algorithm for concatenating two doubly linked lists L and M, with header and trailer sentinel nodes, into a single list L.

1.1.8 Problem C-3.10

Describe in detail how to swap two nodes x and y (and not just their contents) in a singly linked list L given references only to x and y. Repeat this exercise for the case when L is a doubly linked list. Which algorithm takes more time?

For a singly linked list: loop through until the nodes before x and y are found, label them x' and y'. Set x' next node to y, and y next to x original next. Set y' next node to x, and x next node to y original next.

For a doubly linked list: do the same thing except time is not needed to loop through and find the previous nodes. Don't forget to set the previous node values.

1.1.9 Problem C-3.11

Describe in detail an algorithm for reversing a singly linked list L using only a constant amount of additional space and not using any recursion.

Define n to be the size of the linked list. Loop `n-1` times with iterator i starting at 0, and bring node `n-i` backwards by swapping it `n-i` times with the node previous. Save the previous node as a pointer variable.

1.1.10 Problem C-3.22

Suppose you are given two circularly linked lists, L and M, that is, two lists of nodes such that each node has a nonnull next node. Describe a fast algorithm for telling if L and M are really the same list of nodes but with different (cursor) starting points.

Create two node pointers, lowercase l and m, and set them to point to the head of L and M respectively. Loop through the list until l is equal to M and m is equal to L (they are the same list), or l is equal to L and m is equal to M (the whole list has been parsed and L and M are not the same list).

1.1.11 Problem C-3.23

Given a circularly linked list L containing an even number of nodes, describe how to split L into two circularly linked lists of half the size.

Given a circularly linked list L, define M to be the length of L divided by two. Create two node pointers, x and y, and have them point to the beginning of L and the node at M, respectively. For the last node in the first half (M-1), set its next node to the first node. For the first node in the first half (0), set its previous node to M-1. For node M, set its previous node to the last node in the list. Set the last node's next to M.

2 Projects

2.1 Encryption Cipher

Name:	Nicolas Nytko
Course:	CSC216
Activity:	Encryption Cipher
Level:	5
Description:	Write a program that can perform encryption and decryption using an arbitrary substitution cipher. In this case, the encryption array is a random shuffling of the letters in the alphabet. Your program should generate a random encryption array, its corresponding decryption array, and use these to encode and decode a message. Allow for the saving and loading of encrypted messages by storing the 26 letter encryption key amongst the n encoded characters of the message. Note that for an n character message, there will be $n + 1$ slots amongst them. For a 1-character message, for instance, there is a slot before the character and a slot after the character. For a 2-character message, there are slots before and after the first character and after the second character. And so on... Make sure the extra 26 mod $(n + 1)$ letters from the encryption key are located carefully to make the spread nice and even. (Note that when the message is longer than 25 characters, each letter of the key is alone by itself and, in fact, you are spreading the characters of the message amongst the 27 slots around the key values now.)

2.1.1 Compiler Environment

Listing 4: environment

```
   tex git:(master)    pwd
/Users/nicolas/Git/portfolio1/tex
   tex git:(master)    uname -a
Darwin Nicolass-MacBook-Pro.local 16.0.0 Darwin Kernel Version 16.0.0: Mon Aug 29
    17:56:20 PDT 2016; root:xnu-3789.1.32~3/RELEASE_X86_64 x86_64
   tex git:(master)    clang --version
Apple LLVM version 8.0.0 (clang-800.0.38)
Target: x86_64-apple-darwin16.0.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```

2.1.2 Source

Listing 5: ../project/cipher/Makefile

```
CC=g++
OUTPUT=cipher.out
INPUT=main.cpp
CCFLAGS=

all:
	$(CC) $(CCFLAGS) -std=c++14 $(INPUT) -o $(OUTPUT)
```

Listing 6: ../project/cipher/main.cpp

```
#include <iostream>
#include <cmath>
#include <fstream>
#include <cstring>
#include <ctime>
```

```

class CipherKey
{
private:
    static const unsigned long KEY_LENGTH = 26;
    char pKey[KEY_LENGTH + 1]; // Plus one for the null terminator

    unsigned long getKeyIndex( char c ) const
    {
        unsigned long nReturn = 0;

        if ( !( c >= 'A' && c <= 'Z' ) )
            return 0;

        for ( unsigned long i=0; i < KEY_LENGTH && nReturn == 0; i++ )
        {
            if ( c == pKey[i] )
                nReturn = i;
        }

        return nReturn;
    }

public:
    CipherKey( )
    {
        std::memset( pKey, 0, KEY_LENGTH + 1 );
    }

    CipherKey( const CipherKey& pOther )
    {
        std::memcpy( pKey, pOther.pKey, KEY_LENGTH );
    }

    CipherKey& operator=( const CipherKey& pOther )
    {
        std::memcpy( pKey, pOther.pKey, KEY_LENGTH );

        return *this;
    }

    void generate( unsigned int nSeed )
    {
        std::srand( nSeed );

        for ( unsigned long i=0; i < KEY_LENGTH; i++ )
        {
            pKey[i] = 'A' + static_cast<char>( i );
        }

        for ( unsigned long i=0; i < KEY_LENGTH; i++ )
        {
            std::swap( pKey[i], pKey[ static_cast<unsigned int>( std::rand( ) ) %
                KEY_LENGTH ] );
        }
    }

    void setKey( const char* pNewKey )
    {
        if ( strlen( pNewKey ) != KEY_LENGTH )
        {
            std::cerr << "CipherKey::setKey(" << pNewKey << "): new key is not "
                      << KEY_LENGTH << "characters long." << std::endl;
            return;
        }

        std::strncpy( pKey, pNewKey, KEY_LENGTH + 1 );
    }

    std::string encrypt( const std::string& sEncrypt ) const
    {
        std::string sReturn;

        for ( size_t i=0; i < sEncrypt.length( ); i++ )
        {
            char c = sEncrypt[i];

            if ( std::islower( c ) )
                c = static_cast<char>( std::toupper( c ) );

            if ( std::isupper( c ) )
                sReturn += pKey[ c - 'A' ];
        }

        return sReturn;
    }

    std::string decrypt( const std::string& sDecrypt ) const
    {
        std::string sReturn;

        for ( size_t i=0; i < sDecrypt.length( ); i++ )
        {
            unsigned long nIndex = getKeyIndex( sDecrypt[i] );

            sReturn += 'A' + static_cast<char>( nIndex );
        }

        return sReturn;
    }

    unsigned int getKeyLength( ) const
    {
        return KEY_LENGTH;
    }

    const char* getKey( ) const
    {
        return pKey;
    }
};

class FileCipher
{
private:
    CipherKey pCipherKey;
    std::string sMessage;

```

```
public:
    FileCipher( ): pCipherKey( ), sMessage( "" )
    {
        pCipherKey.generate( static_cast<unsigned int>( std::time( NULL ) ) );
    }

    FileCipher( const std::string sSetMessage ): pCipherKey( ), sMessage(
        sSetMessage )
    {
        pCipherKey.generate( static_cast<unsigned int>( std::time( NULL ) ) );
    }

    FileCipher( const FileCipher& pOther ): pCipherKey( pOther.pCipherKey ),
                                            sMessage( pOther.sMessage ) { }

    FileCipher& operator=( const FileCipher& pOther )
    {
        pCipherKey = pOther.pCipherKey;
        sMessage = pOther.sMessage;

        return *this;
    }

    std::string getMessage( ) const
    {
        return sMessage;
    }

    void setMessage( std::string sNewMessage )
    {
        sMessage = sNewMessage;
    }

    std::string encrypt( ) const
    {
        return pCipherKey.encrypt( sMessage );
    }

    std::string decrypt( std::string sToDecrypt )
    {
        return pCipherKey.decrypt( sToDecrypt );
    }

    std::string encryptWithKey( ) const
    {
        std::string sEncrypted = encrypt( );
        std::string sReturn = "";
        const char* szKey = pCipherKey.getKey( );

        double nCharsPerSlot = (double) pCipherKey.getKeyLength( ) /
            ( sMessage.length( ) + 1 );
        unsigned long nCurrentKey = 0;

        /* Do first key pass */

        for ( unsigned long i=nCurrentKey; i < nCharsPerSlot; i++, nCurrentKey++ )
        {
            sReturn += szKey[ nCurrentKey ];
        }

        /* Write all characters */

        for ( unsigned long nChar=0; nChar < sMessage.length( ); nChar++ )
        {
            sReturn += sEncrypted[nChar];

            /* Write keys after all message characters */

            for ( ; nCurrentKey < static_cast<unsigned long>( nCharsPerSlot * (
                nChar + 2 ) ); nCurrentKey++ )
            {
                sReturn += szKey[ nCurrentKey ];
            }
        }

        /* Write any leftover keys */

        for ( ; nCurrentKey < pCipherKey.getKeyLength( ); nCurrentKey++ )
        {
            sReturn += szKey[ nCurrentKey ];
        }

        return sReturn;
    }

    std::string decryptWithKey( std::string sEncrypted )
    {
        std::string sMsg, sKey;

        unsigned long nChars = sEncrypted.length( ) - static_cast<unsigned long>(
            pCipherKey.getKeyLength( ) );
        double nCharsPerSlot = (double) pCipherKey.getKeyLength( ) /
            ( nChars + 1 );

        /* nCurrent is position in overall string, nCurrentKey is
         * current part of the key */

        unsigned long nCurrent=0, nCurrentKey = 0;

        /* Do first key pass */

        for ( ; nCurrentKey < nCharsPerSlot; nCurrentKey++ )
        {
            sKey += sEncrypted[nCurrent++];
        }

        /* Look at all characters */

        for ( size_t nMsg = 0; nMsg < nChars; nMsg++ )
        {
            sMsg += sEncrypted[nCurrent++];

            for ( ; nCurrentKey < static_cast<unsigned long>( nCharsPerSlot * ( nMsg
                + 2 ) ); nCurrentKey++ )
            {
                sKey += sEncrypted[nCurrent++];
            }
        }

        /* Check any leftover keys */

        for ( ; nCurrent < sEncrypted.length( ); nCurrent++ )
        {
            sKey += sEncrypted[nCurrent];
        }

        pCipherKey.setKey( sKey.c_str( ) );
        sMessage = decrypt( sMsg );

        return sMessage;
    }

    friend std::ostream& operator<<( std::ostream& pOutput, const FileCipher&
        pCipher )
    {
        pOutput << pCipher.encryptWithKey( ) << std::endl;

        return pOutput;
    }

    friend std::istream& operator>>( std::istream& pInput, FileCipher& pCipher )
    {
        std::string sLine;
        std::getline( pInput, sLine );

        pCipher.decryptWithKey( sLine );

        return pInput;
    }
};

void printUsage( char* arg0 )
{
    std::cout << "USAGE: " << arg0 << " -f <filename> [options]" << std::endl << std
        ::endl;
    std::cout << "OPTIONS: " << std::endl;
    std::cout << "    -f, --file\t\t\tFile to do operations on." << std::endl;
    std::cout << "    -e, --encrypt <msg>\t\tEncrypts <msg> and saves it in file."
        << std::endl;
    std::cout << "    -d, --decrypt\t\tDecrypts the message loaded from file." <<
        std::endl << std::endl;
}

enum ProgramStatus
{
    STATUS_UNKNOWN,
    STATUS_ENCRYPT,
    STATUS_DECRYPT
};

int main( int argc, char** argv )
{
    std::string sFile;
    std::string sInputMsg;
    ProgramStatus nStatus = STATUS_UNKNOWN;

    if ( argc < 2 )
    {
        printUsage( argv[0] );
        return 1;
    }
    else
    {
        bool bArgsGood = true;
        int i = 1;

        while ( i < argc && bArgsGood )
        {
            char* szArg = argv[i];

            if ( szArg[0] != '-' )
            {
                bArgsGood = false;
            }
            else
            {
                if ( szArg[1] == 'f' || std::strcmp( szArg, "--file" ) == 0 )
                {
                    sFile = argv[i+1];
                    i += 2;
                }
                if ( szArg[1] == 'e' || std::strcmp( szArg, "--encrypt" ) == 0 )
                {
                    nStatus = STATUS_ENCRYPT;
                    sInputMsg = argv[i+1];

                    i += 2;
                }
                if ( szArg[1] == 'd' || std::strcmp( szArg, "--decrypt" ) == 0 )
                {
                    nStatus = STATUS_DECRYPT;
                    i++;
                }
            }
        }

        if ( !bArgsGood )
        {
            printUsage( argv[0] );
            return 1;
        }
    }

    FileCipher pCipher;

    if ( nStatus == STATUS_ENCRYPT )
    {
        std::ofstream fOutput;

        pCipher.setMessage( sInputMsg );
        fOutput.open( sFile, std::ofstream::out );

        fOutput << pCipher;

        fOutput.close( );
    }
    else if ( nStatus == STATUS_DECRYPT )
    {
        std::ifstream fInput;

        fInput.open( sFile, std::ifstream::in );
        fInput >> pCipher;

        std::cout << pCipher.getMessage( ) << std::endl;
    }

    return 0;
}
```

2.1.3 Compiler Output

Listing 7: ../project/cipher/compilerout

```
     cipher git:(master)     make CC=harper_cpp
harper_cpp  -std=c++14 main.cpp -o cipher.out
main.cpp***
```

2.1.4 Program Output

Listing 8: ../project/cipher/progout

```
     cipher git:(master)     ./cipher.out
USAGE: ./cipher.out -f <filename> [options]

OPTIONS:
    -f, --file              File to do operations on.
    -e, --encrypt <msg>         Encrypts <msg> and saves it in file.
    -d, --decrypt         Decrypts the message loaded from file.

     cipher git:(master)     ./cipher.out -f file -e helloworld
     cipher git:(master)     ./cipher.out -f file -d
HELLOWORLD
     cipher git:(master)     ./cipher.out -f file -e goodmorningjasonjames
     cipher git:(master)     ./cipher.out -f file -d
GOODMORNINGJASONJAMES
```

3 Labs

3.1 Game Highscores

Name:	Nicolas Nytko	Course: CSC216
Activity:	Game Highscores	
Level:	4	
Description:	P-3.4. Store 10 game highscores in a doubly-linked list.	

3.1.1 Compiler Environment

Listing 9: environment

```
   tex git:(master)   pwd
/Users/nicolas/Git/portfolio1/tex
   tex git:(master)   uname -a
Darwin Nicolass-MacBook-Pro.local 16.0.0 Darwin Kernel Version 16.0.0: Mon Aug 29 17:56:20 PDT 2016; root:xnu-3789.1.32~3/RELEASE_X86_64 x86_64
   tex git:(master)   clang --version
Apple LLVM version 8.0.0 (clang-800.0.38)
Target: x86_64-apple-darwin16.0.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```

3.1.2 Source

Listing 10: ../lab/highscores/main.cpp

```cpp
#include <iostream>

/* P-3.4
 * Write a class that maintains the top 10 scores for a game application,
 * implementing the add and remove functions of Section 3.1.1, but use a
 * doubly linked list. Your implementation of remove(i) should make the
 * fewest number of pointer hops to get to the game entry at index i.
 */

/**
 * High score entry containing name and score.
 */

class GameEntry
{
private:
    std::string name;
    int score;

public:
    GameEntry( const std::string& setName="", int setScore=0 ): name( setName ), score( setScore ) { }
    GameEntry( const GameEntry& pOther ): name( pOther.name ), score( pOther.score ) { }

    GameEntry& operator=( const GameEntry& pOther )
    {
        name = pOther.name;
        score = pOther.score;

        return *this;
    }

    std::string getName( ) const
    {
        return name;
    }

    int getScore( ) const
    {
        return score;
    }

    void setName( const std::string& sNewName )
    {
        name = sNewName;
    }

    void setScore( int nNewScore )
    {
        score = nNewScore;
    }
};

/**
 * Linked-list node.
 */

class ListNode
{
private:
    GameEntry pData;
    class ListNode* pPrev, *pNext;

public:
    ListNode( ): pPrev( NULL ), pNext( NULL ) { }

    ListNode( const GameEntry& pSetData ): pData( pSetData ),
                                           pPrev( NULL ),
                                           pNext( NULL ) { }

    ListNode( const ListNode& pOther ): pData( pOther.pData ),
                                        pPrev( pOther.pPrev ),
                                        pNext( pOther.pNext ) { }

    ListNode& operator=( const ListNode& pOther )
    {
        pData = pOther.pData;
        pPrev = pOther.pPrev;
        pNext = pOther.pNext;

        return *this;
    }

    GameEntry& getData( )
    {
        return pData;
    }

    GameEntry getData( ) const
    {
        return pData;
    }

    friend class List;
};

/**
 * Linked-list implementation class.
 */

class List
{
private:
    ListNode* pFirst, *pLast;
    size_t nLength;

    ListNode* getNode( size_t nIndex )
    {
        if ( nLength == 0 )
            return NULL;

        ListNode* pReturn;

        if ( nIndex < nLength/2 )
        {
            pReturn = pFirst;

            for ( size_t i=0; i < nIndex; i++ )
            {
                pReturn = pReturn->pNext;
            }
        }
        else
        {
            pReturn = pLast;

            for ( size_t i=nLength-1; i > nIndex; i-- )
            {
                pReturn = pReturn->pPrev;
            }
        }

        return pReturn;
    }

public:
    List( ): pFirst( NULL ), pLast( NULL ), nLength( 0 )
    {
        pFirst = new ListNode;
        pLast = pFirst;
    }

    List( const List& pOther ): pFirst( pOther.pFirst ),
                                pLast( pOther.pLast ),
                                nLength( pOther.nLength ) { }

    List& operator=( const List& pOther )
    {
        pFirst = pOther.pFirst;
        pLast = pOther.pLast;
        nLength = pOther.nLength;

        return *this;
    }

    ~List( )
    {
        if ( nLength != 0 )
        {
            for ( ListNode* pCurrent = pFirst;
                  pCurrent != NULL;
                  pCurrent = pCurrent->pNext )
            {
                delete pCurrent;
            }
        }
    }

    size_t getLength( ) { return nLength; }

    GameEntry& get( size_t nIndex )
    {
        return getNode( nIndex )->getData( );
    }

    GameEntry& operator[]( const size_t nIndex )
    {
        return getNode( nIndex )->getData( );
    }

    void push_back( const GameEntry& pData )
    {
        if ( nLength == 0 )
        {
            pFirst = new ListNode( pData );
            pLast = pFirst;
        }
        else
        {
            ListNode* pTempNode = new ListNode( pData );
            pLast->pNext = pTempNode;
            pTempNode->pPrev = pLast;
            pLast = pTempNode;
        }

        nLength++;
    }

    bool insert( size_t nIndex, const GameEntry& pData )
    {
        if ( nLength == 0 )
        {
            if ( nIndex != 0 )
                return false;

            /* If theres no other nodes, call our push_back function */

            push_back( pData );

            return true;
        }
        else
        {
            if ( nIndex > nLength )
                return false; /* Fail if last node slot + 1 */

            if ( nIndex == nLength )
            {
                /* If we're trying to place at the last slot then call push_back */

                push_back( pData );
                return true;
            }
            else if ( nIndex == 0 )
            {
                /* If we're trying to place at the beginning */

                ListNode* pSecond = pFirst;
                ListNode* pNewTemp = new ListNode( pData );

                pNewTemp->pNext = pSecond;
                pFirst = pNewTemp;

                nLength++;

                return true;
            }
            else
            {
                /* Placing at an arbitrary point in the list */

                ListNode* pAt = getNode( nIndex );
                ListNode* pPrev = pAt->pPrev;
                ListNode* pNewTemp = new ListNode( pData );

                pNewTemp->pPrev = pPrev;
                pNewTemp->pNext = pAt;
                pAt->pPrev = pNewTemp;
                pPrev->pNext = pNewTemp;

                nLength++;

                return true;
            }
        }

        return false;
    }

    bool remove( size_t nIndex )
    {
        if ( nLength == 0 )
        {
```

```cpp
            /* Can't remove when there's already nothing */

            return true;
        }
        else if ( nLength == 1 )
        {
            delete pFirst;
            pFirst = NULL;
            pLast = NULL;

            return true;
        }
        else
        {
            if ( nIndex >= nLength )
                return false;

            /* If last node, remove and update the new last node */

            if ( nIndex == nLength - 1 )
            {
                ListNode* pNewEnd = pLast->pPrev;
                delete pLast;

                pLast = pNewEnd;
                pNewEnd->pNext = NULL;

                return true;
            }
            else
            {
                /* Else, remove node and update next and previous nodes to point
                   to each other */

                ListNode* pPrev, *pNext, *pCurrent;
                pCurrent = getNode( nIndex );
                pPrev = pCurrent->pPrev;
                pNext = pCurrent->pNext;

                delete pCurrent;
                pPrev->pNext = pNext;
                pNext->pPrev = pPrev;

                return true;
            }
        }
    }
};

int main( )
{
    List scores;

    scores.push_back( GameEntry( "BOB", 50000 ) );
    scores.push_back( GameEntry( "NIK", 42000 ) );
    scores.push_back( GameEntry( "ASK", 36900 ) );
    scores.push_back( GameEntry( "TUT", 31000 ) );
    scores.push_back( GameEntry( "DAN", 20000 ) );

    scores.insert( 0, GameEntry( "LOL", 65000 ) );
    scores.insert( 6, GameEntry( "SUX", 100 ) );
    scores.insert( 6, GameEntry( "BWA", 500 ) );
    scores.insert( 6, GameEntry( "ASD", 1000 ) );
    scores.insert( 6, GameEntry( "DSF", 2000 ) );

    for ( size_t i=0; i < scores.getLength( ); i++ )
    {
        std::cout << i << ": " << scores[i].getName( ) << ": " << scores[i].getScore
            ( ) << std::endl;
    }

    return 0;
}
```

3.1.3 Compiler Output

Listing 11: ../lab/highscores/compilerout

```
      highscores  git:(master)      make CC=harper_cpp
harper_cpp  -std=c++14 main.cpp -o highscores.out
main.cpp***
```

3.1.4 Program Output

Listing 12: ../lab/highscores/progout

```
      highscores  git:(master)      ./highscores.out
0: LOL: 65000
1: BOB: 50000
2: NIK: 42000
3: ASK: 36900
4: TUT: 31000
5: DAN: 20000
6: DSF: 2000
7: ASD: 1000
8: BWA: 500
9: SUX: 100
```

3.2 Matrix Class

Name: Nicolas Nytko Course: CSC216

Activity: Matrix Class
Level: 3
Description: P-3.2. Matrix class with multiplication and addition operators.

3.2.1 Compiler Environment

Listing 13: environment

```
      tex  git:(master)      pwd
/Users/nicolas/Git/portfolio1/tex
      tex  git:(master)      uname -a
```

```
Darwin Nicolass-MacBook-Pro.local 16.0.0 Darwin Kernel Version 16.0.0: Mon Aug 29 17:56:20 PDT 2016; root:xnu-3789.1.32~3/RELEASE_X86_64 x86_64
    tex git:(master)    clang --version
Apple LLVM version 8.0.0 (clang-800.0.38)
Target: x86_64-apple-darwin16.0.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```

3.2.2 Source

Listing 14: ../lab/matrix/main.cpp

```cpp
#include <iostream>
#include "matrix.hpp"

int main( )
{
    Matrix<3,2> a = { 1, 2, 3, 4, 5, 6 };
    Matrix<2,3> b = { 7, 8, 9, 10, 11, 12, 13 };
    Matrix<2,3> c = { 14, 15, 16, 17, 18, 19 };

    std::cout << "Matrices: " << std::endl;
    std::cout << "a:   " << a << std::endl << "b:   " << b << std::endl << "c:   " << c << std::endl << std::endl;

    auto product = a * b;
    auto sum = b + c;

    std::cout << "a*b: " << product << std::endl;
    std::cout << "b+c: " << sum << std::endl;

    return 0;
}
```

Listing 15: ../lab/matrix/matrix.hpp

```cpp
#ifndef LIB_MATRIX_HPP
#define LIB_MATRIX_HPP

#include <fstream>

/**
 * Template class for an nRows by nCols matrix.
 */

template <short nRows, short nCols>
class Matrix
{
private:
    double dData[nRows*nCols];

public:
    Matrix( )
    {
        for ( size_t i=0; i < nRows*nCols; i++ )
        {
            dData[i] = 0.0;
        }
    }

    Matrix( std::initializer_list<double> dList )
    {
        for ( size_t i=0; i < dList.size( ) && i < nRows*nCols; i++ )
        {
            dData[i] = *( dList.begin( ) + i );
        }
    }

    constexpr unsigned long getArea( ) const { return nRows*nCols; }
    constexpr short getRows( ) const { return nRows; }
    constexpr short getCols( ) const { return nCols; }
    constexpr short getHeight( ) const { return nRows; }
    constexpr short getWidth( ) const { return nCols; }

    double& operator[]( const size_t nIndex )
    {
        if ( nIndex >= getArea( ) )
            return dData[ getArea( ) - 1 ];

        return dData[ nIndex ];
    }

    double& get( const size_t x, const size_t y )
    {
        return dData[ ( y * nCols ) + x ];
    }

    double get( const size_t x, const size_t y ) const
    {
        return dData[ ( y * nCols ) + x ];
    }

    Matrix<nRows,nCols> add( const Matrix<nRows,nCols>& mOther )
    {
        Matrix<nRows,nCols> mReturn;

        for ( size_t i=0; i < getArea( ); i++ )
        {
            mReturn.dData[i] = dData[i] + mOther.dData[i];
        }

        return mReturn;
    }

    Matrix<nRows,nCols> operator+( const Matrix<nRows,nCols> & mOther )
    {
        return add( mOther );
    }

    template<short nCols2>
    Matrix<nRows,nCols2> multiply( const Matrix<nCols,nCols2>& mOther )
    {
        /* Adapted from pseudocode on https://en.wikipedia.org/wiki/
           Matrix_multiplication_algorithm */

        Matrix<nRows,nCols2> mReturn;

        for ( size_t i=0; i < nRows; i++ )
        {
            for ( size_t j=0; j < nCols2; j++ )
            {
                double dSum = 0;

                for ( size_t k=0; k < nCols; k++ )
                {
                    dSum += get( k, i ) * mOther.get( j, k );
                }

                mReturn.get( j, i ) = dSum;
            }
        }

        return mReturn;
    }

    template<short nCols2>
    Matrix<nRows,nCols2> operator*( const Matrix<nCols,nCols2>& mOther )
    {
        return multiply( mOther );
    }

    friend std::ostream& operator<<( std::ostream& stream, const Matrix& mMatrix )
    {
        stream << "(" << nRows << "x" << nCols << ")[";

        for ( size_t i=0; i < mMatrix.getArea( ); i++ )
        {
            stream << ' ' << mMatrix.dData[i];

            if ( ( i + 1 ) % nCols == 0 && i != mMatrix.getArea( )-1 )
                stream << ',';
            else
                stream << ' ';
        }

        stream << "]";

        return stream;
    }
};

#endif
```

3.2.3 Compiler Output

Listing 16: ../lab/matrix/compilerout

```
     matrix git:(master)     make CC=harper_cpp
harper_cpp  -std=c++14 main.cpp -o matrix.out
main.cpp***
```

3.2.4 Program Output

Listing 17: ../lab/matrix/progout

```
      matrix  git:(master)       ./matrix.out
Matrices:
a:    (3x2)[  1   2,  3   4,  5   6  ]
b:    (2x3)[  7   8   9,  10   11   12  ]
c:    (2x3)[  14   15   16,  17   18   19  ]

a*b:  (3x3)[  27   30   33,  61   68   75,  95   106   117  ]
b+c:  (2x3)[  21   23   25,  27   29   31  ]
```

3.3 Sort Int Array

Name:	Nicolas Nytko	Course: CSC216
Activity:	Sort Int Array	
Level:	3	
Description:	C-3.18. Write a short recursive C++ function that will rearrange an array of int values so that all the even values appear before the odd values.	

3.3.1 Compiler Environment

Listing 18: environment

```
      tex  git:(master)       pwd
/Users/nicolas/Git/portfolio1/tex
      tex  git:(master)       uname -a
Darwin Nicolass-MacBook-Pro.local 16.0.0 Darwin Kernel Version 16.0.0: Mon Aug 29 17:56:20 PDT 2016; root:xnu-3789.1.32~3/RELEASE_X86_64 x86_64
      tex  git:(master)       clang --version
Apple LLVM version 8.0.0 (clang-800.0.38)
Target: x86_64-apple-darwin16.0.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```

3.3.2 Source

Listing 19: ../lab/sortint/main.cpp

```cpp
#include <iostream>
#include <utility>

/* Recursively rearrange an array so that all even numbers
 * will appear before odd ones.  Rearranged array will not
 * be sorted and is not stable */

void sort( int* pArray, size_t nLength )
{
    long nEven = -1, nOdd = -1;

    for ( long i=0; i < static_cast<long>( nLength ) && ( nEven == -1 || nOdd == -1 ); i++ )
    {
        if ( nEven == -1 && pArray[i] % 2 == 0 )
            nEven = i;

        if ( nOdd == -1 && pArray[i] % 2 == 1 )
            nOdd = i;
    }

    if ( nEven != -1 && nOdd != -1 )
    {
        std::swap( pArray[nEven], pArray[nOdd] );
    }

    if ( nLength > 1 )
    {
        sort( pArray + 1, nLength - 1 );
    }
}

template<typename T>
void printArray( T* pArray, size_t nLength )
{
    for ( size_t i=0; i < nLength; i++ )
    {
        std::cout << pArray[i] << ' ';
    }

    std::cout << std::endl;
}

int main( )
{
    const int LENGTH = 20;
    int example[LENGTH] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                            11, 12, 13, 14, 15, 16, 17, 18, 19, 20};

    std::cout << "Array pre sorting:" << std::endl;
    printArray( example, LENGTH );

    sort( example, LENGTH );

    std::cout << "Array post sorting:" << std::endl;
    printArray( example, LENGTH );

    return 0;
}
```

3.3.3 Compiler Output

Listing 20: ../lab/sortint/compilerout

```
     sortint git:(master)     make CC=harper_cpp
harper_cpp  -std=c++14 main.cpp -o sort.out
main.cpp***
```

3.3.4 Program Output

Listing 21: ../lab/sortint/progout

```
     sortint git:(master)     ./sort.out
Array pre sorting:
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Array post sorting:
2 4 6 8 10 12 14 16 18 20 11 3 13 7 15 1 17 9 19 5
```