

ECE 231 - Intermediate Programming and Engineering
Problem Solving
EXAM I - March 24, 2011

87

NAME: _____

Yes calculators and ONE sheet (8X11) of paper containing class related material. No book or class notes. [] Indicates possible point values for the problem.

1. [10 points] The following statement defines a macro that computes the absolute value of its argument: `#define abs(x) ((x) > 0 ? (x) : -(x))`
- a) If $a=1.2$, $b=4.5$, and c are float variables then the following statement appearing in the C/C++ code `c = abs(a-b)` is expanded to what? What is the value of c ? b) What are the advantages/disadvantages of using macros?

2. [10] What is the main purpose of the following statements?

```
#ifndef MYHEADER
#define MYHEADER
/* contents of myheader.h */
#endif
```

3. [10] First identify the problem in the for loop and fix it. Explain what the problem is.

```
for (double x = 0; x != 5.0; x+= 0.1)
{
    y =f(x);
    cout << "x = " << x << "\t y = " << y << endl;
}
```

4. [10 points] Using 8-bit unsigned integer values give examples of overflow and underflow? How can these problems be corrected?

5. [15 points] What does the following code do, explain? What will be displayed at each COUT? If there are problems fix it.

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

int main()
{
    int m = 1;
    for (int i = 0; i < 3; i++) {
        m += m << 0;
    }
    cout << "The value of m is " << m << endl;
}
```

6. [15 points] What does the following code do, explain? What will be displayed at each COUT? If there are problems fix it.

```
#include <iostream>
#define PI 0.14159 + 3.0
using namespace std;

int main() // function main begins program execution
{
    int *p, num;
    float twopi;

    p = &num;
    *p = 505;
    cout << num << '\t';
    (*p)++;
    cout << num << '\t';
    (*p)--;
    cout << num << '\n';
    twopi = 2 * PI;
    cout << "twopi = " << twopi << '\n';

    return 0; // indicate that program ended successfully
} // end function main
```

Nicolas Lopez, ECE 231

7. [15 points] What does the following code do, explain? What will be displayed at each COUT ?

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

int main() // function main begins program execution
{
    char str[] = "This is a test!";
    char *start, *end;
    int len;
    char t;

    cout << "Original: " << str << "\n";

    len = strlen(str);

    start = str;
    end = &str[len-1];

    while (start < end )
    {
        //swap chars
        t = *start;
        *start = *end;
        *end = t;

        // advance pointers
        start++;
        end--;
    }
    cout << str << "\n";
    return 0; // indicate that program ended successfully
} // end function main
```

8. [15 points] Given the code below and the starting address 0x7fff5fbff95c, determine what will be printed out, explain.

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

int main()
{
    int i = 98765,
        j = 12345;
    double d = 3.14,
           e = 2.86;
        // Declare pointer variables that:
    int * iPtr, // store addresses of ints
        * jPtr;
    double * dPtr, // store addresses of doubles
           * ePtr;
    iPtr = &i; // value of iPtr is address of i
    jPtr = &j; // value of jPtr is address of j
    dPtr = &d; // value of dPtr is address of d
    ePtr = &e; // value of ePtr is address of e

    cout << "\nAt address " << iPtr
         << ", the value " << *iPtr+20 << " is stored.\n"
         << "\nAt address " << jPtr
         << ", the value " << *jPtr-10 << " is stored.\n"
         << "\nAt address " << dPtr
         << ", the value " << dPtr*9 << " is stored.\n"
         << "\nAt address " << ePtr
         << ", the value " << *ePtr/2 << " is stored.\n";
}
```

1. a. $c = \text{abs}(a-b)$

$c = ((a-b) > 0 ? (a-b) : -(a-b))$

$c = ((-3.3) > 0 ? (-3.3) : -(-3.3))$

$c = (-3.3) : -(-3.3)$

$c = 3.3$

b. This will negate any negative number, but it will attempt to negate the number zero. $\rightarrow 2$

Less overhead, faster to execute.

2. This statement will ensure that the contents of my header.h avoid redclaration errors. $\rightarrow 2$

3. Loop should be changed to $x <= 5.0$,
for (double $x=0$; $x <= 5.0$; $x += 0.1$)

4. $255 \quad 11111111$
 $+ 1 \quad + 00000001$
 $256 \quad 100000000$

9 bit overflow, cannot represent

256 in 8 bits

$0 \quad 00000000$
 $- 256 \quad + 00000001 \quad 2^8$
 $- 256 \quad 00000001$
 11111111

underflow, cannot represent as fix

5. The code outputs powers of two as determined by the exit condition for the loop

for (int i = 0; i < (3 * (K+1)))

3 true because here $n = 2^n$
is no shift $m = 8$

6. The code creates a variable "int num" and a pointer to it "*p". The value is set to 505. The code displays num then uses the pointer "*p" to change the value in memory and num changes accordingly.

509

506

505

3.28318

The last part is supposed to multiply our definition of pi by two (2 * 3.14159). However, the definition is improper and functions as $[(2 * 14159) + 3]$
The definition should be PI (0.14159 + 3)

7. The code creates two char strings, one of which is initialized as "This is a test!". The first cout displays "Original" and all characters stored in the initialized string. The while loop starts with the first char of the initialized string and stores it at the end of the second until it is reversed and copied entirely. The last cout displays this reversed string

8. At 0x7fff5fbff95c ... 98785 is stored
At 0x7fff5fbff958 ... 12335 is stored
At 0x7fff5fbff950 ... 28.26 is stored
At 0x7fff5fbff948 ... 1.43 is stored

As the size of int is 4 bytes, the address of j is 4 spots in memory down, in hex form. Also as size of double is 8 bytes ptrs d and e are 8 spots below the ptr they come after

In the second part of each cont, the $*ptr$ is used to reference the values stored and operated on before displaying

HW2, HW3, HWS

ECE 251 - Intermediate Programming and Engineering
Problem Solving
EXAM I - March 24, 2011

100

NAME: _____

Yes calculators and ONE sheet (8X11) of paper containing class related material. No book or class notes. [] Indicates possible point values for the problem.

1. [10 points] The following statement defines a macro that computes the absolute value of its argument: #define abs(x) ((x) > 0 ? (x) : -(x))

a) If a=1.2, b=4.5, and c are float variables then the following statement appearing in the C/C++ code c = abs(a-b) is expanded to what? What is the value of c? b) What are the advantages/disadvantages of using macros?

$c = ((1.2 - 4.5) > 0 ? (1.2 - 4.5) : -(1.2 - 4.5)) = -(-3.3) \Rightarrow c = 3.3$

macros don't have any overhead and make it easy to modify constants, but it is easy to use parenthesis incorrectly, resulting in unexpected results.

2. [10] What is the main purpose of the following statements?

```
#ifndef MYHEADER
#define MYHEADER
/* contents of myheader.h */
#endif
```

This prevents the same declarations from being carried out multiple times. If the code has already been read, MYHEADER is defined and it is ignored.

avoid redeclaration errors

3. [10] First identify the problem in the for loop and fix it. Explain what the problem is.

```
for (double x = 0; x  $\neq$  5.0; x += 0.1)
```

this should be <=, as currently written the loop would immediately exit

```
{
    y = f(x);
    cout << "x = " << x << "\t y = " << y << endl;
}
```

4. [10 points] Using 8-bit unsigned integer values give examples of overflow and underflow? How can these problems be corrected?

Overflow and underflow occur when a number is too large or small to be stored with a fixed number of bits

using $x = 11111111$, $y = 00000010$

$x+y$ would produce 10000001 , which cannot be stored in 8 bits.

$z = x+y$ would wrap around and return 00000001 .

unsigned int's cannot store negative values, so $00000001 - 00000010$ will wrap around to the highest possible value and return 11111111 .

5. [15 points] What does the following code do, explain? What will be displayed at each `COUT`? If there are problems fix it.

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

```
int main()
```

```
{
    int m = 1;
    for (int i = 0; i < 3; i++) {
        m += m << 0;
    }
    cout << "The value of m is " << m << endl;
}
```

The code adds m to itself three times and outputs the result. As correctly written, it will output 7.

The value of m is 8

The shift operator <<, does not do anything when using

0. If given a non-zero number, it will shift all the bits of m that number of places to the left.

6. [15 points] What does the following code do, explain? What will be displayed at each `COUT`? If there are problems fix it.

```
#include <jstream>
#define PI(0.14159 + 3.0)
using namespace std;
```

```
int main() // function main begins program execution
```

```
{
    int *p, num;
    float twopi;

    p = &num;
    *p = 505;
    cout << num << "\n";
    (*p)++;
    cout << num << "\n";
    (*p)--;
    cout << num << "\n";
    twopi = 2 * PI;
    cout << "twopi = " << twopi << "\n";
}
```

This code declares a pointer, links it to a float, addresses the pointer to increment and decrement the float. This produces the output:

505 506 505

The second output was the ~~defining~~ statement. To properly calculate `twopi`, this statement ~~misses~~ has parenthesis added around the two numbers so that when it is used, operators act on the sum of the two numbers.

As currently written, this would output:

twopi = 3.28318

with parenthesis added, the output is:

twopi = 6.28318

```
return 0; // indicate that program ended successfully
} // end-function main
```

7. [15 points] What does the following code do, explain? What will be displayed at each COUT ?

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

```
using namespace std;
```

```
int main() // function main begins program execution
{
```

```
    char str[] = "This is a test!";
    char *start, *end;
    int len;
    char t;
```

```
    cout << "Original: " << str << "\n";
```

```
    len = strlen(str);
```

```
    start = str;
    end = &str[len-1];
```

```
    while (start < end)
    {
```

```
        //swap chars
```

```
        t = *start;
        *start = *end;
        *end = t;
```

This program reverses the string declared as char str[].

The first cout will produce:

Original: This is a test!

and the second will produce:

!tset a si sihT

Notes: the code given with the examples has different numeric values in the print statements, and starts with a different address. This description is for the code on this page.

8. [15 points] Given the code below and the starting address 0x7fff5fbff95c, determine what will be printed out, explain.

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

int main()
{
    int i = 98765,
        j = 12345;
    double d = 3.14,
           e = 2.86;

    // Declare pointer variables that:
    int * iPtr, // store addresses of ints
        * jPtr;
    double * dPtr, // store addresses of doubles
           * ePtr;

    iPtr = &i; // value of iPtr is address of i
    jPtr = &j; // value of jPtr is address of j
    dPtr = &d; // value of dPtr is address of d
    ePtr = &e; // value of ePtr is address of e

    cout << "\nAt address " << iPtr
         << ", the value " << *iPtr+20 << " is stored.\n"
         << "\nAt address " << jPtr
         << ", the value " << *jPtr-10 << " is stored.\n"
         << "\nAt address " << dPtr
         << ", the value " << dPtr*9 << " is stored.\n"
         << "\nAt address " << ePtr
         << ", the value " << *ePtr/2 << " is stored.\n";
}
```

glanfuntio of the
 This code will print out the two integer and two double values along with their addresses.
 Integers are stored as 4 bytes and doubles as 8, so the addresses will decrease accordingly.

This will print:

- At address 0x7fff5fbff95c, the value 98785 is stored.
- At address 0x7fff5fbff958, the value 12335 is stored.
- At address 0x7fff5fbff950, the value 28.26 is stored.
- At address 0x7fff5fbff948, the value 1.43 is stored.

Because the main is performed to the print statement, the result is not stored at the given locations, and the printed statements are incorrect.

ECE 231 - Intermediate Programming and Engineering
Problem Solving
EXAM I - March 24, 2011

50

NAME: _____

Yes calculators and ONE sheet (8X11) of paper containing class related material. No book or class notes. [] Indicates possible point values for the problem.

1. [10 points] The following statement defines a macro that computes the absolute value of its argument: `#define abs(x) ((x) > 0 ? (x) : -(x))`

a) If $a=1.2$, $b=4.5$, and c are float variables then the following statement appearing in the C/C++ code $c = \text{abs}(a-b)$ is expanded to what? What is the value of c ? b) What are the advantages/disadvantages of using macros?

a) $c = |1.2 - 4.5| = -2.3$ 3.3

Faster, less overhead -2

b) The advantage of using macros is that they set boundaries around problems making them easier to solve; however, the disadvantage is that only users understand them.

2. [10] What is the main purpose of the following statements?

```
#ifndef MYHEADER
#define MYHEADER
/* contents of myheader.h */
#endif
```

These statements define the folders of source files of the contents located under (or in) MYHEADER

avoid redeclaration errors -2

3. [10] First identify the problem in the for loop and fix it. Explain what the problem is.

```
for (double x = 0; x != 5.0; x+= 0.1)
{
    y = f(x);
    cout << "x = " << x << "\t y = " << y << endl;
}
```

4. [10 points] Using 8-bit unsigned integer values give examples of overflow and underflow? How can these problems be corrected?

Overflow occurs when memory of computer code uses more system resources than what is possible. 1000 is an example of overflow. Underflow is when memory or code is less than the maximum boundaries.

These problems can be corrected by running or averaging the results of an underflow and overflow value. -5

5. [15 points] What does the following code do, explain? What will be displayed at each COUT? If there are problems fix it.

```
#include <iostream> <ostream.h>
using namespace std;
```

```
int main()
{
    int m = 1;
    for (int i = 0; i < 3; i++) {
        m += m << 0;
    }
    cout << "The value of m is " << m << endl;
}
```

The code is running three instances of the equation $m += m \ll 0$ starting with $m=1$. This value is displayed as 3 once all three instances of the for loop have run.

$m = 8$?

6. [15 points] What does the following code do, explain? What will be displayed at each COUT? If there are problems fix it.

```
#include <iostream> <iostream.h>
#define PI 0.14159 + 3.0
using namespace std;
```

```
int main() // function main begins program execution
{
    int *p, num;
    float twopi;

    p = &num;
    *p = 505;
    cout << num << '\n';
    (*p)++;
    cout << num << '\n';
    (*p)--;
    cout << num << '\n';
    twopi = 2 * PI;
    cout << "twopi = " << twopi << '\n';

    return 0; // indicate that program ended successfully
} // end function main
```

The code is running a pointer in the file to look for the definition of the numeric value of PI address. Once found, this PI value times two.

The int defines *p: num as an integer value. The float defines twopi (the value of PI multiplied by 2) as a number within a certain range of values.

505, 506, 505

3.28318

7. [15 points] What does the following code do, explain? What will be displayed at each COUT ?

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

using namespace std;

int main() // function main begins program execution
{
    char str[] = "This is a test!";
    char *start, *end;
    int len;
    char t;

    cout << "Original: " << str << "\n";

    len = strlen(str);

    start = str;
    end = &str[len-1];

    while (start < end )
    {
        //swap chars
        t = *start;
        *start = *end;
        *end = t;

        // advance pointers
        start++;
        end--;
    }
    cout << str << "\n";
    return 0; // indicate that program ended successfully
} // end function main
```

8. [15 points] Given the code below and the starting address 0x7fff5fbff95c, determine what will be printed out, explain.

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

int main()
{
    int i = 98765,
        j = 12345;
    double d = 3.14,
           e = 2.86;
    // Declare pointer variables that:
    int * iPtr, // store addresses of ints
        * jPtr;
    double * dPtr, // store addresses of doubles
           * ePtr;
    iPtr = &i; // value of iPtr is address of i
    jPtr = &j; // value of jPtr is address of j
    dPtr = &d; // value of dPtr is address of d
    ePtr = &e; // value of ePtr is address of e

    cout << "\nAt address " << iPtr
         << ", the value " << *iPtr+20 << " is stored.\n"
         << "\nAt address " << jPtr
         << ", the value " << *jPtr-10 << " is stored.\n"
         << "\nAt address " << dPtr
         << ", the value " << dPtr*9 << " is stored.\n"
         << "\nAt address " << ePtr
         << ", the value " << *ePtr/2 << " is stored.\n";
}
```



ECE 231 – Programming Assignment 1– Spring 2011

Tentative due date is Friday, February 4th.

1) Write a C Program to perform complex number, $z=a+jb$, operations such as:

- Real (z)
- Imaginary (z)
- Conjugate (z^*)
- Magnitude (z)
- Phase (z)
- Addition ($z1 + z2$)
- Subtraction ($z1 - z2$)
- Multiplication ($z1 * z2$)
- Division ($z1 / z2$)
- Power (z^n)
- N-th root ($z1/n$)

2) You are welcome to use the following “complex.h” file or use your own.

```
/* File: complex.h */
#ifndef COMPLEXh
#define COMPLEXh
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <math.h>
#define FORMAT "%8.3lf"

typedef enum {cartesian, polar} form;

typedef struct {
    double real, imag;
    form current_form;
} complex;

void print_complex(complex z)
{
    printf("%8.3lf %8.3lf", z.real, z.imag);
}
```



Department of Electrical & Computer Engineering

```
/* Function Prototypes */  
void print_complex(complex);  
complex add(complex, complex);  
complex subtract(complex, complex);  
complex multiply(complex, complex);  
complex divide(complex, complex);  
complex conjugate(complex); // complex conjugate  
complex powerofcomplex(int n-power, complex);  
complex rootofcomplex(int m-root, complex);  
float realpart(complex);  
float imaginarypart(complex);  
float magnitudeofcomplex(complex);  
float phaseofcomplex(complex);
```

```
#endif
```

3) Input/Output specification agreed in class are as follows:

- Your program will read an input file that contains the following information
 - The first variable is an integer that indicates the number of complex numbers to be processed
 - The second integer variable gives you the nth-power of the complex number to be found
 - The third integer variable gives you the m-th-root of the complex number to be found
 - The fourth variable (you decide what it is) tells you the format in which the complex numbers that follow in the file are; that is, if they are in real and imaginary format (Cartesian) or magnitude and phase format (Polar)
 - The rest of the variables that follow in the file are floating point numbers
 - Real part or magnitude of first complex number
 - Imaginary part or phase of first complex number
 - Real part or magnitude of second complex number
 - Imaginary part or phase of second complex number
 -
- The second complex number needed for the add, subtract, multiply and divide operations will be prompted to the user for him/her to enter. NOTE: format must be the same as the complex numbers in the input file.
- The output with all the results will be written to an output file. Make sure you specify the format of the complex number results.



Department of Electrical & Computer Engineering

ECE 231 –Assignment 10 – Spring 2011

Due date: May 3, 2011

Individual work!!!!

You are asked to write a 1 hour and 15 minutes Exam. The questions should be based on Chapters 5 through 8 of the textbook, and Assignments 6 through 8. You also must provide the answers. Keep in mind that this is an exam to be taken in class, closed book and notes, and in 1 hour and 15 minutes. If code needs to be analyzed, debugged, explained, compiled and executed, then it will be emailed to the class at the beginning of the exam. Exam II will be given on Thursday, May 5th, during class. Be prepared to present and discuss your questions, Tuesday, May 3rd, 2011.



Department of Electrical & Computer Engineering

ECE 231 –Assignment 2 – Spring 2011

Due date: Tuesday, February 22, 2011

You are welcome to work in teams of maximum 4 people.

Search the Internet to learn about the following topics and produce a paper that summarizes your findings. I expect to see a lot of references (URLs) and high-level explanation of the following topics.

- How the http protocol works
- Wireshark
- Android phone, hardware architecture and software development kits
- Iphone and ipad, hardware architecture and software development kits
- Autosar
- ARM processor
- Scrum
- Agile
- Software Engineering
- Software estimation techniques – for example Function point analysis
- UML, and tools for UML
- CMM
- OOD
- Ruby
- Ruby on Rails
- Perl
- Python
- Apache
- Xcode, other environments
- P2P communications
- Cloud computing
- Scalability of software
- Hacking
- SKYPE
- IP ports
- Open GL
- Napster
- www.nist.gov



ECE 231 –Assignment 3 – Spring 2011

Due date: March 4, 2011

You are welcome to work in teams of maximum 4 people.

You are asked to develop a C++ program to determine the statistics of an array of floating point numbers. You will read the numbers from a file and write the results also to a file. The statistics to be found are: MAX number, MIN number, mean or average, variance, standard deviation, produce a histogram of the numbers and sort them in ascending or descending order. Use dynamic memory allocation and make sure you release it once you are done processing.

Specifications done in class:

- 1) Use argc and argv[] in main to enter the number of files
- 2) You will have one input file that you will read until detecting end of file and use a counter to determine the number of floating points to process. You will generate a uniform distribution as one input file. The second input file will consist of a Gaussian distribution.
- 3) You will produce three output files: first file will contain the basic statistics like MIN, MAX, mean, variance, standard deviation, number of bins and or bin width for histogram. Second file will be the sorted numbers in ascending order. Third file will have the histogram.
- 4) To plot the histogram you will use MATLAB.

Reference Book: Numerical Recipes in C or C++



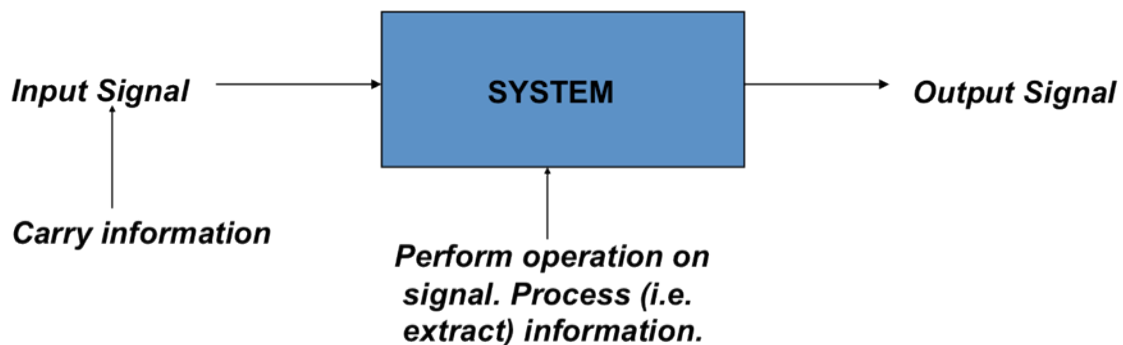
ECE 231 –Assignment 4 – Spring 2011

Due date: March 11, 2011

You are welcome to work in teams of maximum 4 people.

You are asked to develop a C++ program to perform the linear convolution of the input sequence given by $x(n)$ of size N , and the impulse response sequence of the system given by $h(n)$ of size M . Both sequences use floating point numbers. Each sequence is given as an input file and your result should also be written to a file. Use dynamic memory allocation and make sure you release it once you are done processing.

- Input Signal = Input Sequence = $x(n)$; size or duration N ; read file until you detect EOF to determine the size.
- Impulse Response Sequence = $h(n)$; size or duration M ; read file until you detect EOF to determine the size of this file.
- Use `argc` and `argv[]` to execute your program: input and output file names
- Output Signal = Output Sequence = $y(n)$; size or duration $L = N + M - 1$; write result to output file.
- To plot your sequences use MATLAB.



Convolution Sum

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$



Department of Electrical & Computer Engineering

ECE 231 –Assignment 5 – Spring 2011

Due date: March 22, 2011

Individual work!!!!

You are asked to write a 1 hour and 15 minutes Exam. The questions should be based on Chapters 3 and 4 of Dr. Heileman's book, Chapters 1 through 4 of the textbook, and Assignments 1 through 4. You also must provide the answers. Keep in mind that this is an exam to be taken in class, closed book and notes, and in 1 hour and 15 minutes. If code needs to be analyzed, debugged, explained, compiled and executed, then it will be emailed to the class at the beginning of the exam. Exam I will be given on Thursday, March 24th, during class. Be prepared to present and discuss your questions, Tuesday, March 22, 1011.



Department of Electrical & Computer Engineering

ECE 231 –Assignment 6 – Spring 2011

Due date: March 31, 2011
Individual work!!!!

You are asked to rewrite in C++ the Complex Number Arithmetic assignment. In the .h file you should define the Class “231complex” and include all the necessary libraries, constants, structures, etc, etc. A second file will be your driver, and a third file will contain all the functions. You will need to change all the I/O you implemented in C to C++.



ECE 231 – Assignment 7 – Spring 2011

Due date: April 14, 2011

Work in teams of maximum 2 people!!!!

You are asked to write in C++ code that will identify a list of palindrome words, numbers and sentences from an input file. Use strings and linked lists.

Specifications

- 1) Read list of palindrome words, numbers and sentences (strings) from an input file. Use argc and argv parameters for I/O.
- 2) Write to an output file the input list and mark with a YES (or TRUE) or NO (or FALSE) whether each member of the list is palindrome or not.
- 3) For part 2 use the class string operations of C++ to do the processing
- 4) Using linked lists to compose a palindrome word/sentence with all the palindrome strings and write it out to the output file.
- 5) Special characters like blank spaces, punctuation marks, are not to be counted.

Example

Input file contains the following words, numbers, and sentences

```
747
radar
civic
Dogma: I am God
Mouse
```

Output file should contain

```
747 TRUE
radar TRUE
civic TRUE
Dogma: I am God TRUE
Mouse FALSE
```

```
74radcivDogmaIamGodvicdar47
```



ECE 231 – Assignment 8 – Spring 2011

Due date: April 21, 2011

Work in teams of maximum 3 people!!!!

You are asked to write in C++ code that will implement an interactive calculator using Postfix (RPN) notation to perform complex numbers arithmetic. Use dynamic stacks.

Specifications

- 1) The user will enter the numbers and operations to be performed. Thus do not need to use argc and argv parameters for I/O.
- 2) Use your existing complex class and functions to perform the operations.
- 3) Use your existing flag to determine whether you are processing the complex numbers in Cartesian or Polar forms.
- 4) Use strings “power” or “root” to perform the complex operations specified by the strings.
- 5) For testing purposes use redirections commands to input numbers and operations and to produce an output file. Ex: `./rpn < inputfile > outfile`



ECE 231 –Assignment 9 – Spring 2011

Due date: May 5, 2011

Work in teams of maximum 4 people!!!!

You are asked to write in C++ code that will find **all** paths between an Entrance and Exit to a Maze. Also, you should find the optimal path (shortest distance). In the WEBCT site you will find the original maze (maze.pbm) and different solutions (mazeXX.pgm).

Specifications

Initial Maze

- 1) The walls are color black (number 0)
- 2) Entrance and Exit are white (number 1)
- 3) Make a border around the Maze of color white (number 1)

Final Maze(s)

- 1) The walls are color black (number 0)
- 2) Entrance and Exit are the highest number equivalent to the optimal path color white (number 3)
- 3) Suboptimal paths are gray color (number 2)
- 4) Border is color 1, darker gray than suboptimal paths

Generation of Mazes

- 1) Install the software “gimp” or similar. That is, image processing software that allows you to read images in different formats, processes them, and convert to other file formats
- 2) To generate a Maze go to hereandabove.com/maze/mazeorig.form.html
- 3) Create a Maze with path width of 1 pixel and wall width of any size
- 4) Save image it creates in format .gif
- 5) Open the image in gimp and save it as a .pbm file and in ASCII
- 6) Edit the .pbm image and remove the comment line that begins with #; save the image
- 7) .pbm images start with header parameter P1, when you save your images in .pgm make sure you add at beginning of the header the parameter P2