Question 1: [20 points]

A brief, yet interesting history of logarithms for those of you who are interested:

"Logarithms were invented independently by John Napier (Figure 1), a Scotsman, and by Joost Burgi, a Swiss. The logarithms which they invented differed from each other and from the common and natural logarithms now in use. Napier's logarithms were published in 1614; Burgi's logarithms were published in 1620. The objective of both men was to simplify mathematical calculations. Napier's approach was algebraic and Burgi's approach was geometric. Neither men had a concept of a logarithmic base. Napier defined logarithms as a ratio of two distances in a geometric form, as opposed to the current definition of logarithms as exponents. The possibility of defining logarithms as exponents was recognized by John Wallis in 1685 and by Johann Bernoulli in 1694." [1]

Figure 1: Johnny Napier.[2]

For this question, you will be writing a C++ function which will compute the **integer**

$\log_{\textbf{Keith}}(\textbf{Mick})$, which is defined to be the **smallest** integer (call it **Charlie**) such that:

$\textbf{Keith}^{\textbf{Charlie}} >= \textbf{Mick}$.

Note that the numbers **Keith**, **Mick** and **Charlie** are all positive integer quantities. **Mick** is assumed to be greater than 0.

The following table gives the values of the integer $\log_{\text{Keith}}(\text{Mick})$ for several values of the base Keith and for several values of Mick.

Table 1: Some Integer logarithms for 3 different bases

Mick	$\log_{10}(\text{Mick})$	$\log_{2}(\text{Mick})$	$\log_{3}(\text{Mick})$
1	0	0	0
2	1	1	1
3	1	2	1
4	1	2	2
5	1	3	2
6	1	3	2
7	1	3	2
8	1	3	2
9	1	4	2
10	1	4	3
11	2	4	3
12	2	4	3
13	2	4	3
14	2	4	3
15	2	4	3
16	2	4	3
etc.	etc.	etc.	etc.

You **must** use the following function prototype:

```
int integer_log(int keith, int mick )

// keith = base , mick = x
//
// find the smallest integer charlie such that
//
```

```
//       charlie
//  keith          >= mick
```

Use **only** a `while` statement, the multiplication arithmetic operator, the addition operator and any initialization of variables which may be required, as well as the `return` statement within your function. In other words, you are **not** allowed to use the `pow()`, `log()`, or `log10`() functions found in the <**cmath**> library.

The following program typifies how this function would be used within main():

```
// Author: Ted Obuchowicz
// April 21, 2009
// unsigned binary integer log (x)  program
//                              y
// using function

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

int integer_log(int , int  );

int main()
{
int x;  // we restrict x to be greater than 0
int base;

cout << "Please enter the value x you wish to compute the integer log of " ;
cin >> x;

cout << "Please enter the base of the logarithm " ;
cin >> base;

cout << "The answer is: "  << integer_log(base, x) << endl;

return 0;
}
```

Hint: keith ^{0} = 1

keith ^{1} = keith = 1 * keith

keith ^{2} = keith * keith = 1 * keith * keith

keith ^{3} = keith * keith * keith = 1 * keith * keith * keith

keith ^{mick} = keith * keith * keith * ... * keith (keith multiplied by itself mick times)

Another Hint: Figure out how many times you have to multiply 1 by Keith in order for it to become bigger or equal to Mick.

Question 2: [20 points]

Give the output produced by the following program:

```
// Author: Ted Obuchowicz
// April 21, 2009

#include <iostream>
#include <string>

using namespace std;

void mystery(int* p3, int* p2, int* p1, int which_row, int which_col)
{
int temp;
temp = *(p3 + (which_row * 3) + which_col );
*(p3 + (which_row * 3) + which_col ) = *(p2 + (which_row * 3) + which_col );
*(p2 + (which_row * 3) + which_col ) = *(p1 + (which_row * 3) + which_col );
*(p1 + (which_row * 3) + which_col ) = temp;
}

void print(int* p)
{
 for (int row = 0 ; row < 3; row++)
 {
  for(int col = 0 ; col < 3; col++)
    cout <<  *(p + ( row * 3 ) + col )  << " " ;
  cout << endl;
 }
}

int main()
{

int* mick;
int* keith;
int* charlie;

mick    = new int [9];
keith   = new int [9];
charlie = new int [9];


for(int stones = 0 ; stones < 9 ; stones++)
 mick[stones] = stones;

for(int stones = 0 ; stones < 9 ; stones++)
 keith[stones] = stones+9;

for(int stones = 0 ; stones < 9 ; stones++)
 charlie[stones] = stones+18;

cout << "BEFORE: " << endl << endl;
```

```
print(mick);
cout << endl;

print(keith);
cout << endl;

print(charlie);
cout << endl;

mystery(mick, keith, charlie, 0 , 1);

cout << “AFTER: “ << endl << endl;

print(mick);
cout << endl;

print(keith);
cout << endl;

print(charlie);
cout << endl;

return 0;
}
```

Question 3: [20 points]

(a) Write a complete C++ program which opens two text files for input and a binary file used for output. The program is to read a **double** from an **ASCII text** file which is saved with filename “mick_numbers.txt” and a second **double** from another **ASCII text file** (“keith_numbers.txt”). The program is to compute the arithmetic sum of these two numbers, display the result to the screen and save the result (as a **double**) in a **binary** file called “freddy.dat”. The program is to repeatedly read in numbers from the text files until there are no more numbers left to be read.

(b) Assume that the two ASCII text file have the following contents:

mick_numbers.txt:

```
1.1111111111
2.2222222222
3.3333333333
4.4444444444
5.5555555555
```

keith_numbers.txt:

```
1.1111111111
2.2222222222
3.3333333333
4.4444444444
```

```
5.5555555555
```

(i) What would be the file size (in number of bytes) for these two ASCII text files?

(ii) What would the file size (in number of bytes) be for the output file produced by the program?

(c) Suppose we have written a program which opens the binary file "freddy.dat" created by the program in part (a) for input. The program simply reads in a number from this file and displays it to the screen. Such a program is :

```
// Author: Ted Obuchowicz
// April 21, 2009

#include <iostream>
#include <iomanip>
#include <fstream>

using namespace std;

int main()
{

ifstream infile("freddy.dat" , ios_base::binary | ios_base::in);

double  answer;

while ( !infile.eof()  )
{
 infile.read( (char *) &answer, sizeof(double) ) ;
 cout << setprecision(15) << answer << endl;
}

infile.close();

return 0;
}
```

Give a plausible explanation why this program may work on one particular computer, and yet may give bizarre and unexpected results when compiled and run on another type of computer.
For example, if I compile and run the above program on a Solaris Sunblade 100 workstation it produces the following output (which is correct):

```
ted@brownsugar 11:03pm >g++ -o two_files_read_binary two_files_read_binary.C
ted@brownsugar 11:04pm >two_files_read_binary
2.2222222222
4.4444444444
6.6666666666
8.8888888888
11.111111111
```

(Note: brownsugar is the name of my Sun Microsystems SunBlade 100 workstation)

If we were to compile the same program using an IBM PC such as the ones found in the ENCS labs, the program will produce the following bizarre and incorrect results:

```
ted@tux 10:58pm >g++ -o two_files_read_binary two_files_read_binary.C
ted@tux 11:06pm >two_files_read_binary
-2.24860149981593e+281
-2.24860149981725e+281
5.58804582878798e+267
-2.24860149981857e+281
1.87880629299811e-34
```

(Note: tux is the name of an ENCS IBM PC running Linux)

Question 4: [20 points]

(a) Consider the following C++ program which makes use of two classes:

```
// Author: Ted Obuchowicz
//  April 21, 2009

#include <iostream>
using namespace std;

class bad_singer
{
 int my_private_part;
 public:
 bad_singer() {my_private_part = 6;}
 friend class  even_worse_singer ;
 void  look_at_my_own_private();
 void look_at_someone_elses_private(even_worse_singer);
};


class  even_worse_singer
{
 int my_private_part;
 public :
 even_worse_singer() { my_private_part = 12 ;}
 friend  class bad_singer;
 void look_at_my_own_private();
 void look_at_someone_elses_private(bad_singer);
};


void bad_singer::look_at_my_own_private()
{
cout << “My own private part is: “ << my_private_part << endl;
}

void bad_singer::look_at_someone_elses_private(even_worse_singer arg)
{
```

```
cout << “Their private part is: “ << arg.my_private_part << endl;
}


void even_worse_singer::look_at_my_own_private()
{
cout << “My own private part is: “ << my_private_part << endl;
}

void even_worse_singer::look_at_someone_elses_private(bad_singer arg)
{
cout << “Their private part is: “ << arg.my_private_part << endl;
}


int main()
{

bad_singer madonna ;
even_worse_singer britney;

madonna.look_at_my_own_private();
britney.look_at_my_own_private();
madonna.look_at_someone_elses_private(britney);
britney.look_at_someone_elses_private(madonna);

cout << “Good thing Mick is not here to look at any private data ... “ << endl;

return 0;
}
```

The output produced by the program is:

(i)
```
My own private part is: 12
My own private part is: 6
Their private part is: 6
Their private part is: 12
Good thing Mick is not here to look at any private data ...
```

(ii)
```
My own private part is: 6
My own private part is: 12
Their private part is: 12
Their private part is: 6
Good thing Mick is not here to look at any private data ...
```

(iii) No output will be produced as the program will not compile since the C++ language does not allow for mutual friendship between two classes.

(iv)
```
My own private part is: 6
```

```
   My own private part is: 12
   Their private part is: 12
   Their private part is: 6
   My own private part is: 6
   My own private part is: 12
   Their private part is: 12
   Their private part is: 6
   My own private part is: 6
   My own private part is: 12
   Their private part is: 12
   Their private part is: 6
 (repeated forever until a stack overflow occurs)
```

(v) `Who is Mick and why is he looking at private data?`

(vi) `Where is Keith?`

(b) Consider the following program:

```
// Author: Ted Obuchowicz
// April 22, 2009
#include <iostream>
using namespace std;

class that
{
int x;
public:
that() { x = 5;}
void what_is_this();
};

void that::what_is_this()
{
 cout << "This object is stored beginning in memory location: " << that <<
endl;
}

int main()
{
that johnny, be, good;
johnny.what_is_this();
be.what_is_this();
good.what_is_this();
return 0;
}
```

The output produced by the program is:

(i)
```
This object is stored beginning in memory location: 0xffbff99c
This object is stored beginning in memory location: 0xffbff998
This object is stored beginning in memory location: 0xffbff994
```

(Note: the actual values of the memory addresses may vary from the ones given in this sample output).

(ii) no output will be produced as the program will not compile. The Linux g++ compiler will produce the following error message:

```
that.C: In member function ‘void that::what_is_this()’:
that.C:22: error: parse error before ‘<<’ token
```

(iii)
```
This object is stored beginning in memory location: 3214384308
This object is stored beginning in memory location: 3214384304
This object is stored beginning in memory location: 3214384300
```

(Note: the actual values of the memory addresses may vary from the ones given in this sample output).

(iv) No output will be produced as the program compiles without errors but enters an infinite loop when we try to run the program.

Question 5: [20 points]

Consider the following program :

```
// Author: Ted Obuchowicz
// April 22, 2009

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

int a[3] = {2,3,4};

int keith(int mick, int charlie)
{
 int y = 1;

 if ( 0 == charlie )
 {
   return 1;
 } // if
 else
 {
   for(int chuck = 1 ; chuck <= charlie ; chuck++)
```

```
   {
     y = y * mick;
   } // for
   return y;
 } // else
} // end of function keith


int poly(int x, int n)
{
 if ( n == 0 )
   return a[0];
 else
   return (  (a[n] *  keith(x,n))  +  poly(x,n-1)  );
}


int main()
{

cout << poly(2,2) << endl;
return 0;
}
```

(a) Give the output produced by the program.

(b) Draw the function call graph. Use the notation given in Figure 2 when drawing your call graph.

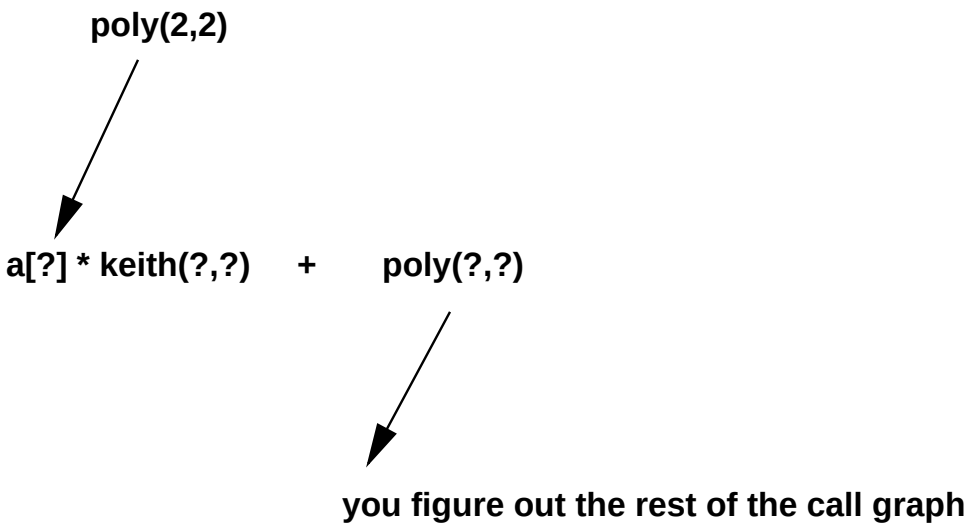

Figure 2: Notation to be employed in drawing the function call graph.

(c) Express in a clear, concise English sentence the task performed by function `keith(mick, charlie)`. An example of how to phrase an answer would be similar to:

"The function keith receives two integer arguments mick and charlie and returns as an integer the value of the mick multiplied by charlie".

Of course, this is only meant as an example, in reality function `keith` does something else.

Do **not** express your answer in the form of:

"Function keith intializes y to 1. Then it checks whether charlie is equal to 0. If it is, then it returns 1, otherwise the function then enters a for loop from chuck equal to 0 up to chuck less than or equal to charlie and chuck is then plused plused. Within the body of the chuck for loop, y is multiplied by mick and the result stored back in y. Upon termination of the chuck for loop, the function returns y."

Answers of this type will receive a grade of **0**.

(d) Express in a clear, concise English sentence the task performed by function `poly(x,n)`. Express your answer using a manner similar to that employed in part (c).

References:

1. http://www.sosmath.com/algebra/logs/log1/log1.html.
2. http://www.fva.is/%7Ebgk/st513/2005h/h5/data/John_Napier.gif