Maths 1264 Quiz 1 answers Thursday 22/1/15

(it is ok to use notes and to collaborate)

On Intel machines, ints and short ints are stored 'little endian,' but in these hand-calculations they should be written in the natural order, with the high-order byte first, short ints as 4 hex digits and ints as 8 hex digits.

(1) convert $(1234)_8$ to decimal and $(1234)_{10}$ to octal (face value, not short int). The first is

$$((1 \times 8 + 2) \times 8 + 3) \times 8 + 4$$

(evaluated as a decimal number).

Answer

```
1234 in octal is 668 in decimal
1234 in decimal is 2322 in octal
```

(2) Given short ints $6f6c$ and $a720$ (hex and 'big endian'), (i) add them and (ii) convert the numbers and the sum to signed decimal.

Answers.

```
Sum is 168c in short
6f6c in short is 28524 in decimal
a720 in short is -22752 in decimal
168c in short is 5772 in decimal
```

(3) (i) Convert 16384 and 16385 from decimal to short int, (ii) add them as short ints, and (iii) convert the result to signed decimal. (iv) What is happening?

```
(i)
16384 is short int 4000 (hex)
16385 is short int 4001 (hex)
(ii)
sum                8001 (hex)
(iii)
This is -32767 in decimal
(iv) Overflow: the sum is out of range
```

(4) Convert -4321 to *32-bit* integer, giving the result as 8 hex digits.

Answer.

```
ff ff ef 1f
```

(5) Decode the following character string, given in hex. It is one string broken over two lines. Character 0 (in C, strings are indexed beginning at 0) is 'H', character 3 is 'h', the last character on the first line is a quote (apostrophe), and the final 00 marks the end of the string.

```
48 65 20 68 69 64 20 6f 6c 64 20 6c 61 64 69 65 73 27
20 72 65 61 64 69 6e 67 20 67 6c 61 73 73 65 73 00
```

Answer.

```
He hid old ladies' reading glasses
```

Maths 1264 Quiz 2 answers Thursday 5/2/15

(it is ok to use notes and to collaborate)

(1: 25 marks) Given the following declarations, on a 32-bit address machine,

```
double x[10];  int a[4][5];
```

(a) How much space is occupied by the array `x`? (not counting the variable `x` itself, which takes 4 bytes)
80 bytes.
(b) And by the array `a`?
80 bytes.
(c) Suppose that `x` begins at location 200, and that `a` is placed immediately after `x`. Where does `a` begin?
Address 280.
(d) What is the address of the last **byte** in `a`?
Address 359.
(e) What is the address of `x[5]`?
Address 240.
(f) What is the *value* (the value is an address) of `a[2]`?
320.
(g) What is the address of `a[3][4]`?
$280 + 3 \times 20 + 4 \times 4 = 336$.
(h) C does not heed array bounds. The address of `x[13]` is actually inside the array `a`. Indeed, it is the address of `a[i][j]` for some `i, j`. Calculate `i` and `j`.

$$\&(x[13]) = 200 + 13 \times 8 = 304$$
$$304 = 280 + 20 \times i + 4 \times j$$
$$304 = 280 + 20 + 4$$
$$i = 1, j = 1$$

Answer `a[1][1]`.

(2: 25 marks) Carefully simulate the following code fragment. (a) Tabulate the values of `i, x, y,` and `z`, and the condition `i < 1`, and (b) show what gets printed.

(c) What does the code print in terms of `n`?

```
int x,y,n,i;

n = 4;
x = y  = 0;
for (i=0; i<n; ++i)
{
  y += 3*i + 3*x + 1;
  x += 2*i + 1;
}
printf("n %d x %d y %d\n", n, x, y);
```

Answers.

(a)

n	i	x	y	'i < n'
4		0	0	
	0			
			y	
		1		
	1			
			y	
		8		
	4			
	2			
			y	
		27		
	9			
	3			
			y	
		64		
	16			
	4			
			n	

(b)prints

n 4 x 16 y 64

(c) prints n, n^2, n^3.

Maths 1264 Quiz 3 ANSWERS Thursday 19/2/15

(it is ok to use notes and to collaborate)

(1) What does the following program print, and why?

```
#include <stdio.h>
int m = 1, n = 2;
int a ( int n )
{
  ++n;
  return n;
}
int b ( int x )
{
  ++n;
  return m+n+x;
}
int c ( int x )
{
  int m=25;
  return m+n+x;
}
main()
{
  int x;

  x = a ( 3 ); printf("x == %d\n", x);
  x = a ( 3 ); printf("x == %d\n", x);
  x = b ( 4 ); printf("x == %d\n", x);
  x = b ( 4 ); printf("x == %d\n", x);
  x = c ( x ); printf("x == %d\n", x);
  x = b ( 4 ); printf("x == %d\n", x);
}
```

Answer...

```
x == 4  a(3) returns 4, changing no globals
x == 4  a(3) returns 4, changing no globals
x == 8  b(4) n 3, m 1, x 4 local: 8
x == 9  b(4) n 4 m 1 x 4 local: 9
x == 38 c(9) m 25 local x 9 local n 4: 38
x == 10 b(4) n 5 m 1 x 4 local: 10
```

(2) Write a function `int is_vowel ( char x )` which returns 1 if `x` is a vowel and 0 otherwise.

(3) Write a function `int vowel_pairs ( char x[] )` which counts the number of adjacent pairs of vowels in `x[]` (assume `x[]` is a valid string). An adjacent pair occurs when $x[i]$ and $x[i+1]$ are equal and are vowels: $x[i+1]$ and $x[i+2]$ could also be an adjacent pair.

Answers.

```
#include <stdio.h>
int is_vowel ( char x )
{
  static char table[10] = "aeiouAEIOU";
  int i;

  for (i=0; i<10; ++i)
    if ( x == table[i] )
      return 1;

  return 0;
}
int vowel_pairs ( char x[] )
{
  int i;
  int count = 0;
  for (i=0; x[i] != '\0'; ++i)
    if ( is_vowel( x[i] ) && x[i] == x[i+1] )
      ++ count;
  return count;
}
```

(4) Suppose that the code below is presented with the following 3-line file

```
We know that you highly esteem the kind of Learning taught in those
Colleges, and that the Maintenance of our young Men, while with
you, would be very expensive to you.  We are convinc'd, therefore,
```

What does it print, and why?

```
#include <stdio.h>
char * line[1000];
void decr ( char * str )
{ // the usual code to remove a newline ...  }
main()
{ int i, n;
  char buffer [ 200 ];
```

```
  n = 0;
  while ( fgets( buffer, 200, stdin ) != NULL )
  {
    decr ( buffer );
    line[n] = buffer;
    ++n;
  }
  for (i=0; i<n; ++i)
    printf("%s\n", line[i]);
}
```

What gets printed:

```
you, would be very expensive to you.  We are convinc'd, therefore,
you, would be very expensive to you.  We are convinc'd, therefore,
you, would be very expensive to you.  We are convinc'd, therefore,
```

Reason: `line[i] == buffer` for $0 \leq i < n$. It prints the *ultimate* contents of the buffer, n times.

(5) Simulate the following (recursive) code, including what gets printed. You should show the call and return pattern of the recursive routine.

What does the function `xxx( m, n )` compute in general, given $n \geq 0$?

```
#include <stdio.h>
int xxx ( int m, int n )
{
  int x;
  if ( n == 0 )
    return 0;
  x = xxx ( m, n/2 );
  if ( n % 2 == 0 )
    return x + x;
  else
    return x + x + m;
}
main ()
{
  printf("xxx(8,9) is %d\n", xxx(8,9));
}
```

```
Answer.
main
  xxx x 8 9
  | xxx x 8 4
  | | xxx x 8 2
  | | | xxx x 8 1
```

```
  | | | | xxx x 8 0
  | | | | return 0
  | | | xxx x 8 1 resume
  | | | |   0
  | | | |return 8
  | | xxx x 8 2 resume
  | | |   8
  | | |return 16
  | xxx x 8 4 resume
  | |  16
  | |return 32
  xxx x 8 9 resume
  |  32
  |return 72
prints
xxx(8,9) is 72
```

In general, it computes $m \times n$.

Maths 1264 Quiz 4 ANSWERS Thursday 12/3/15

(it is ok to use notes and to collaborate)

(1: 15 marks) Show *exactly* what the following C program prints, and explain why.

```
#include <stdio.h>
main()
{
  int x = 123, i,j,k;
  char d[] = "0123456789abcdef";
  unsigned char * xx = (char *) & x;
  for (i=0; i<sizeof(int); ++i )
  {
    j = xx[i] / 16; k = xx[i] % 16;
    printf("%c%c", d[j], d[k]);
  }
  printf("\n");
}
```

Answer. It prints

```
7b000000
```

Rationale. It treats the integer x as a character string, taking each byte and printing a pair of hex digits. Converting 123 to `int` produces $0000007b$ (hex). The output is little endian.

(2: 13 marks) Given

```
typedef class Vec3
{ public:
```

```
    Vec3();
    Vec3(double, double, double);
    Vec3 operator+(const Vec3 & other);
    void print ( );
  private:
    double x1, x2, x3;
} Vec3;
```

write code for

```
Vec3 Vec3::operator + (const Vec3 & other)
```

Answer.

```
Vec3 Vec3::operator + (const Vec3 & other)
{
  return Vec3 ( x1 + other.x1, x2 + other.x2, x3 + other.x3 );
}
```

(3)(i: 5 marks) What is the rationale for `const Vec3 & other`?
Answer. It saves space and time, in that the full `Vec3` is not copied. The difference is trivial in this case.
(ii: 5 marks) Given

```
#include <string>
...
  char x[20];
  string y;
...
  cin >> x;
  cin >> y;
```

Both input statements read the next word — sequence of nonblanks surrounded by blanks — from the input. One of them is ok and the other is ill-advised. Which is which, and why?
Answer `cin >> y` is safe, because the string is made as long as necessary to take all the data. Reading into `x` could cause memory to be overwritten, which is dangerous.
(4: 12 marks) Say *exactly* what the following program does, statement-by-statement in the main procedure, and why. **Note.** The increment operators $++x$ and $x++$ both change x, but they have different *values*: the value of $++x$ is the value *after* incrementing and $x++$ *before*.

```
#include <iostream>
using namespace std;
int n = 0;

void a ( int n )
{
  cout <<  "a " << ++n << endl;
}
```

```
void a ( double n )
{
  cout <<  "a " << ++n << endl;
}

void b ( int & n )
{
  cout <<  "b " << n++ << endl;
}
void c ( int & n )
{
  cout <<  "c " << ++n << endl;
}
int main ()
{
  int n=5;
  double x = 6.5;
  a(n); b(n); a(x);
  b(n); b(n);
  c(n); c(n);
}
```

```
Answer.
n = 5, x = 6.5, a(n) increments call-by-value n and prints:
a 6
b(n) prints n and increments call-by-reference
b 5   and n is 6
a(x) increments call-by-value x and prints
a 7.5
b(n) prints call-by-reference n and increments
b 6 and n is 7
b(n) again
b 7 and n is 8
c(n)  increments and prints call-by-reference n
c 9  and n is 10
c(n) again
c 10  (and n is 11)
```

Maths 1264 Quiz 5 ANSWERS Thursday 26/3/15

(it is ok to use notes and to collaborate)

(1: 15) This is supposed to be a *complete* C++ program. Spot all the mistakes, and correct them.

```
void increment ( int n )
{ ++ n;
```

```
}

int main()
{ int n = 0;
  while ( n < 10 );
    increment(n);

  cout >> "finished\n";

  return 0;
}

Corrected:
#include <iostream>
using namespace std;
void increment ( int & n )
{
  ++ n;
}

int main()
{
  int n = 0;
  while ( n < 10 )
    increment(n);

  cout << "finished\n";

  return 0;
}
```

(2: 20) People's records are stored as follows — 8-digit id, name:

```
14279203 Polk, James K.
14836146 Taylor, Zachary
14984574 Fillmore, Millard
14769015 Pierce, Franklin
```

Write a C++ STL program to read in this data, storing sets of names and ids and maps from names to ids and from ids to names, then to print the results in name order, and then in ID order, as follows:

```
Fillmore, Millard ID: 14984574
Pierce, Franklin ID: 14769015
Polk, James K. ID: 14279203
```

```
Taylor, Zachary ID: 14836146

14279203 NAME: Polk, James K.
14769015 NAME: Pierce, Franklin
14836146 NAME: Taylor, Zachary
14984574 NAME: Fillmore, Millard

#include <iostream>
#include <set>
#include <map>
#include <string>

using namespace std;


int main()
{
  char buffer[200];
  string str;
  int i;
  string id, name;

  set<string> names, ids;
  map<string, string> by_name, by_id;

  while ( cin.getline( buffer, 200 ) )
  {
    str = string ( buffer );
    i = str.find_first_of (' ');
    id = str.substr ( 0, i );
    name = str.substr ( i+1 );

    names.insert(name);
    ids.insert (id);

    by_id[id] = name;
    by_name[name] = id;
  }

  for ( set<string>::iterator it = names.begin();
        it != names.end(); ++it )
  {
    cout << *it << " ID: " << by_name[*it] << endl;
  }
```

```
  cout << endl;

  for ( set<string>::iterator it = ids.begin();
        it != ids.end(); ++it )
  {
    cout << *it << " NAME: " << by_id[*it] << endl;
  }

  return 0;
}
```

(3: 15) Carefully simulate the following code, showing what it prints. What does `xxx(n)` compute, for general nonnegative n?

```
#include <stdio.h>
int xxx ( int n )
{ int x, y, z;
  if ( n < 11 )
    return n;
  else
  { x = n % 12;
    y = xxx ( n/12 );
    z = x + y;
    if ( z > 10 )
      z -= 11;
    return z;
  }
}
main( int argc, char * argv[] )
{ printf("xxx(1234) is %d\n", xxx(1234));
}
```

continued...

```
(Experimenting with another layout, where the stack frames
are not indented.)
main
                          |
  n    x    y    z       |
                          | xxx(1234)
1234  10                 |
                          |  | xxx(102)
 102   6                 |  |
                          |  |  | xxx ( 8 )
                          |  |  return 8
```

```
 102   6                  |  | resume
            8             |  |
                14        |  |
                 3        |  | return 3
1234  10    3             | resume
                13        |
                 2        | return 2
main prints:
xxx(1234) is 2
```

In general, `xxx(n)` computes $n\%11$, if $n \geq 0$.