

IC210 Introduction to Computer Science  
12-Week Exam, Fall 2008 (AY09)

**STOP!**

Read and initial each of the directions below.

DJS You may fill out this cover sheet when you receive your exam copy, but you may not start on the exam or look at the questions until told to begin by your instructor.

DJS Unless otherwise specified, you are to assume that all C++ source code given in this exam is syntactically correct.

DJS When writing code, use **call-by-reference ONLY** when **necessary**.

DJS This exam is closed book / closed notes. Electronic devices are not permitted.

Name: ANSWER KEY Alpha: \_\_\_\_\_

Page	Max Points	Score
2.	10	
3.	20	
4.	10	
5.	20	
6.	20	
7.	20	
Total	100	

1. (10 pts) Do not try to make practical sense of the code below as it serves no purpose other than for you to demonstrate your understanding of program flow-of-control, scope, and parameter passing.

Show the output of this program.

```
int main( )
{
    int a = 1, b = 2;
    cout << "main:" << a << "-" << b << endl;
    fizz( a, b );
    cout << "main:" << a << "-" << b << endl;
    bazz( b, a );
    cout << "main:" << a << "-" << b << endl;

    return 0;
}

void fizz( int b, int &a )
{
    cout << "fizz:" << a << "-" << b << endl;
    if( b < a ) b = a;
    else      a = b;
    cout << "fizz:" << a << "-" << b << endl;
}

void bazz( int &a, int b )
{
    cout << "bazz:" << a << "-" << b << endl;
    fizz( b, a );
    cout << "bazz:" << a << "-" << b << endl;
}
```

Output:

```
main:1-2
fizz:2-1
fizz:2-2
main:1-2
bazz:2-1
fizz:2-1
fizz:2-2
bazz:2-1
main:1-2
```

2. (10 pts) Complete the definition of a recursive function named **oddDigitsRev** that prints, in reverse order, only the odd digits of its integer argument. For example:

```
oddDigitsRev( 123456 ) outputs: 531
oddDigitsRev( 987654 ) outputs: 579
oddDigitsRev( 12 )    outputs: 1
oddDigitsRev( 0 )     outputs: (there is no output)
```

```
void oddDigitsRev( int n )
{

    if( n == 0 ) return;

    if( n%2 == 1 ) cout << n%10;

    oddDigitsRev( n/10 );

}
```

3. (10 pts) Complete the definition of a recursive function named **evenDigits** that prints only the even digits of its integer argument. For example:

```
evenDigits( 123456 ) outputs: 246
evenDigits( 987654 ) outputs: 864
evenDigits( 12 )    outputs: 2
evenDigits( 1 )     outputs: (there is no output)
```

```
void evenDigits( int n )
{

    if( n == 0 ) return;

    evenDigits( n/10 );

    if( n%2 == 0 ) cout << n%10;

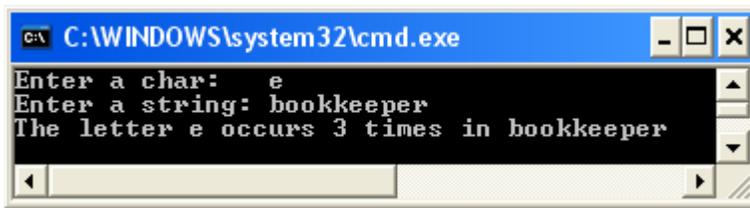
}
```

4. A function named **occursIn** counts the number of times a particular character appears in a C++ string, and returns that count to the caller.

a. (5 pts) Write the function prototype for **occursIn**:

```
int occursIn( char c, string s );
```

b. (5 pts) Assume function **occursIn** has been defined. Complete the code below using a function call to **occursIn** to produce the indicated output:



```
char c;
string s;
cout << "Enter a char:  "; cin >> c;
cout << "Enter a string: " ; cin >> s;
// your code below:

cout << "The letter " << c << " occurs " << occursIn(c,s) << " times in " << s;
```

5. Here's code that finds the average of all the integer values in a data file.

a. (10 pts) Complete the parts that say "your code here"

```

ifstream fin( "data.txt" );
if( fin.fail() ) return 1;

int size;
fin >> size;

// Declare and dynamically allocate memory for the array A:
// your code here:

int *A = new int[size];

double sum;
// -----
sum = 0.0;
for( int i = 0; i < size; i++ )
{
    fin >> A[i];
    sum = sum + A[i];
}
// -----

fin.close();

cout << "average is " << sum / size << endl;

// De-allocate array A's memory:
// your code here:

delete [] A;
    
```

b. (5 pts) A function named **sum1Darray** returns the sum of the elements in a one-dimensional integer array A of size N. Write the prototype for this function:

```
int sum1Darray( int A[], int N );
```

c. (5 pts) Show here how you would call the function named **sum1Darray** *in place of* the code in part a delineated by the dashed lines, that calculates the sum.

```
sum = sum1Darray( A, size );
```

6. (16 pts) Complete the following table. If a value is an address but the address is not known, enter ADDR. The first one is done for you:

```
char n[32] = "WT-Door";
char *a = n;
char *p = new char;
char *s = new char[32];
```

Expression	type	value
n	char *	&n[0]
a	char *	&n[0]
p	char *	ADDR
s	char *	ADDR
n[1]	char	'T'
a[6]	char	'r'
*p = n[0]	char	'W'
&s	char **	ADDR
s[0] = a[0]	char	'W'

7. Write the prototype for each of these functions:

a. (4 pts) A function named **encrypt** returns a C++ string that is an encrypted version of its argument. Write its prototype.

```
Example use:      cout << "Enter a string to encrypt:";
                  string s;
                  cin >> s;
                  cout << "Enter an integer as an encryption key: ";
                  int k;
                  cin >> k;
                  cout << encrypt( s, k );
```

Prototype: string encrypt( string s, int k );

7b. (4 pts) A function named **split** takes an existing array, sorts it, and dynamically creates two new arrays. The new arrays contain the lower and upper halves of the sorted original array, split at the median value. The function returns the median value. Write its prototype.

```
Example use:      double A[11] = { 4, 6, 10, 3, 1, 9, 11, 5, 2, 7, 8 };
                  double *lo, *hi;
                  double median = split( A, 11, lo, hi );
                  cout << "Median is " << median << endl;
                  cout << "Lower half is: ";
                  for( int i = 0; i < (11/2); i++ )
                     cout << lo[i] << ' ';
```

```
Output would be: Median is 6.
                  Lower half is: 1 2 3 4 5
```

Prototype: double split( double \*A, int N, double\* &lo, double\* &hi );

8. (16 pts) Write the definition of a function named **decrementArray** that subtracts one from every element in an array of integers named **A** having size **N**. The function returns **true** or **false** depending on whether any element value, after the subtraction, is less than zero. It also has a parameter named **count** that will hold the total number of elements that became exactly zero due to the subtraction. Here is an example use:

```
int A[1000], count;
// array A is filled somehow, here.
if( decrementArray( A, 1000, count ) != true )
    cout << "none are negative, ";
cout << count << " are zero";

bool decrementArray( int A[], int N, int &count )
{
    bool flag = false;
    count = 0;
    for( i = 0; i < N; i++ )
    {
        A[i] = A[i] - 1;
        if( A[i] < 0 ) flag = true;
        if( A[i] == 0 ) count++;
    }
    return flag;
}
```