IC210: Introduction to Computing

Fall AY2016 – 12-Week Exam

Individual work. Closed book. Closed notes. You may not use any electronic device. This is a multi section exam that will be given to different midshipmen at different times. As per USAINST 1531.53A, you may NOT communicate about this exam with anyone using any medium until your instructor tells you that you can.

Name: ___________________________ , Alpha: _______ , Section Number: _______

Instructor name: ___________________________

<table>
<thead>
<tr>
<th>Dec Hex Char</th>
<th>Dec Hex Char</th>
<th>Dec Hex Char</th>
<th>Dec Hex Char</th>
<th>Dec Hex Char</th>
<th>Dec Hex Char</th>
<th>Dec Hex Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 20</td>
<td>46 2e</td>
<td>60 3c</td>
<td>74 4a J</td>
<td>88 58 X</td>
<td>102 66 f</td>
<td>116 74 t</td>
</tr>
<tr>
<td>33 21 !</td>
<td>47 2f /</td>
<td>61 3d =</td>
<td>75 4b K</td>
<td>89 59 Y</td>
<td>103 67 g</td>
<td>117 75 u</td>
</tr>
<tr>
<td>34 22 &quot;</td>
<td>48 30 0</td>
<td>62 3e &gt;</td>
<td>76 4c L</td>
<td>90 5a Z</td>
<td>104 68 h</td>
<td>118 76 v</td>
</tr>
<tr>
<td>35 23 #</td>
<td>49 31 1</td>
<td>63 3f ?</td>
<td>77 4d M</td>
<td>91 5b [</td>
<td>105 69 i</td>
<td>119 77 w</td>
</tr>
<tr>
<td>36 24 $</td>
<td>50 32 2</td>
<td>64 40 @</td>
<td>78 4e N</td>
<td>92 5c \</td>
<td>106 6a j</td>
<td>120 78 x</td>
</tr>
<tr>
<td>37 25 %</td>
<td>51 33 3</td>
<td>65 41 A</td>
<td>79 4f O</td>
<td>93 5d )</td>
<td>107 6b k</td>
<td>121 79 y</td>
</tr>
<tr>
<td>38 26 &amp;</td>
<td>52 34 4</td>
<td>66 42 B</td>
<td>80 50 P</td>
<td>94 5e ^</td>
<td>108 6c l</td>
<td>122 7a z</td>
</tr>
<tr>
<td>39 27 '</td>
<td>53 35 5</td>
<td>67 43 C</td>
<td>81 51 Q</td>
<td>95 5f –</td>
<td>109 6d m</td>
<td>123 7b {</td>
</tr>
<tr>
<td>40 28 (</td>
<td>54 36 6</td>
<td>68 44 D</td>
<td>82 52 R</td>
<td>96 60 =</td>
<td>110 6e n</td>
<td>124 7c</td>
</tr>
<tr>
<td>41 29 )</td>
<td>55 37 7</td>
<td>69 45 E</td>
<td>83 53 S</td>
<td>97 61 a</td>
<td>111 6f o</td>
<td>125 7d }</td>
</tr>
<tr>
<td>42 2a *</td>
<td>56 38 8</td>
<td>70 46 F</td>
<td>84 54 T</td>
<td>98 62 b</td>
<td>112 70 p</td>
<td>126 7e ~</td>
</tr>
<tr>
<td>43 2b +</td>
<td>57 39 9</td>
<td>71 47 G</td>
<td>85 55 U</td>
<td>99 63 c</td>
<td>113 71 q</td>
<td>127 7f –</td>
</tr>
<tr>
<td>44 2c ,</td>
<td>58 3a :</td>
<td>72 48 H</td>
<td>86 56 V</td>
<td>100 64 d</td>
<td>114 72 r</td>
<td>128 7g &amp;</td>
</tr>
<tr>
<td>45 2d -</td>
<td>59 3b ;</td>
<td>73 49 I</td>
<td>87 57 W</td>
<td>101 65 e</td>
<td>115 73 s</td>
<td>129 7h -</td>
</tr>
</tbody>
</table>

Operator Name | Associativity | Operators |
--------------|---------------|-----------|
Primary scope resolution | left to right | :: |
Primary | left to right | () [ ] . -> dynamic_cast typeid |
Unary | right to left | ++ -- + - ! ~ & * (type_name) sizeof new delete |
C++ Pointer to Member | left to right | .*->* |
Multiplicative | left to right | * / % |
Additive | left to right | + - |
Bitwise Shift | left to right | << >> |
Relational | left to right | < > <= >= |
Equality | left to right | == != |
Bitwise AND | left to right | & |
Bitwise Exclusive OR | left to right | ^ |
Bitwise Inclusive OR | left to right | |
Logical AND | left to right | && |
Logical OR | left to right | || |
Conditional | right to left | ?: |
Assignment | right to left | += -= *= /= <<= >>= %= &= ^= |= |
Comma | left to right | , |

<table>
<thead>
<tr>
<th>Operator Name</th>
<th>Assiociativity</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary scope resolution</td>
<td>left to right</td>
<td>::</td>
</tr>
<tr>
<td>Primary</td>
<td>left to right</td>
<td>() [ ] . -&gt; dynamic_cast typeid</td>
</tr>
<tr>
<td>Unary</td>
<td>right to left</td>
<td>++ -- + - ! ~ &amp; * (type_name) sizeof new delete</td>
</tr>
<tr>
<td>C++ Pointer to Member</td>
<td>left to right</td>
<td>.<em>-&gt;</em></td>
</tr>
<tr>
<td>Multiplicative</td>
<td>left to right</td>
<td>* / %</td>
</tr>
<tr>
<td>Additive</td>
<td>left to right</td>
<td>+ -</td>
</tr>
<tr>
<td>Bitwise Shift</td>
<td>left to right</td>
<td>&lt;&lt; &gt;&gt;</td>
</tr>
<tr>
<td>Relational</td>
<td>left to right</td>
<td>&lt; &gt; &lt;= &gt;=</td>
</tr>
<tr>
<td>Equality</td>
<td>left to right</td>
<td>== !=</td>
</tr>
<tr>
<td>Bitwise AND</td>
<td>left to right</td>
<td>&amp;</td>
</tr>
<tr>
<td>Bitwise Exclusive OR</td>
<td>left to right</td>
<td>^</td>
</tr>
<tr>
<td>Bitwise Inclusive OR</td>
<td>left to right</td>
<td></td>
</tr>
<tr>
<td>Logical AND</td>
<td>left to right</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td>Logical OR</td>
<td>left to right</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>right to left</td>
<td>?:</td>
</tr>
<tr>
<td>Assignment</td>
<td>right to left</td>
<td>+= -= *= /= &lt;&lt;= &gt;&gt;= %= &amp;= ^=</td>
</tr>
<tr>
<td>Comma</td>
<td>left to right</td>
<td>,</td>
</tr>
</tbody>
</table>
1. [11pts] In the following code, clearly identify (e.g. circle and label) every
   1. function prototype
   2. function definition
   3. function call
   4. function argument
   5. function parameter

   Note: function fabs() is defined in cmath with prototype double fabs(double);

   ```cpp
   #include <cmath>
   #include <iostream>
   #include <fstream>
   using namespace std;
   
   int rating2int(string s);
   
   string int2rating(int k);
   
   bool similar(int r1, int r2);
   
   int main()
   {
      int* R = new int[20];
      string* N = new string[20];
      string s;
      ifstream fin("bonds.txt");
      for(int i = 0; i < 20; i++)
      {
         fin >> s >> N[i];
         R[i] = rating2int(s);
      }
      cin >> s;
      int t = rating2int(s);
      int k = 0;
      while(k < 20 && !similar(t,R[k]))
         k++;
      cout << int2rating(R[k]) << " " << N[k] << endl;
      return 0;
   }
   
   int rating2int(string s)
   {
      return 3*(s[0] - 'A') + 3 - s.length();
   }
   
   string int2rating(int k)
   {
      string res = "";
      char c = k/3 + 'A';
      for(int i = 0; i < 3 - k % 3; i++)
         res += c;
      return res;
   }
   
   bool similar(int r1, int r2)
   {
      return fabs(r1-r2) <= 1;
   }
   ```
2. [18pts] Write the type for each underlined expression.
   Note: function fabs() is defined in cmath with prototype double fabs(double);

   #include <cmath>
   #include <iostream>
   #include <fstream>
   using namespace std;

   int rating2int(string s);
   string int2rating(int k);
   bool similar(int r1, int r2);

   int main()
   {
      int* R = new int[20];
      string* N = new string[20];
      string s;
      ifstream fin("bonds.txt");
      for(int i = 0; i < 20; i++)
      {
         fin >> s >> N[i];
         R[i] = rating2int(s);
      }
      cin >> s;
      int t = rating2int(s);
      int k = 0;
      while(k < 20 && !similar(t, R[k]))
      {
         k++;
         cout << int2rating(R[k]) << " " << N[k] << endl;
      }
      return 0;
   }

   int rating2int(string s)
   {
      return 3*(s[0] - 'A') + 3 - s.length();
   }

   string int2rating(int k)
   {
      string res = "";
      char c = k/3 + 'A';
      for(int i = 0; i < 3 - k % 3; i++)
      {
         res += c;
      }
      return res;
   }

   bool similar(int r1, int r2)
   {
      return fabs(r1-r2) <= 1;
   }
3. [12pts]
   a. Write the code (as it would appear in main(), for example) that creates a variable A and allocates and initializes so that we have the situation depicted in the picture to the right.

   ![Diagram of matrix A]

   b. Write a complete definition of function `printcol` that could be called like this `printcol(A, 4, 5, 1)` (assuming the above situation) to print the index 1 column of the array A. Thus, the call `printcol(A, 4, 5, 1)` would produce the output shown below. Note: the function should work for any 2D array of ints with proper dimensions and column number.

   ```
   2
   7
   12
   17
   ```
4. [12pts] Below a program I'd like to have. Write down the prototypes (not definitions) you would need for each of the functions called.

Note: Not that it really matters for this problem, but this program should read strings from data.txt and store them in the array d, then reorder the strings from shortest to longest, then print out all the strings containing "hap" (from shortest to longest).

```cpp
int main()
{
    ifstream fin("data.txt");
    int n;
    string* d = read(fin,n);
    reorder(d,n);
    int k = -1;
    while((k = indexOfNextContain(d,n,"hap",k)) < n)
        cout << d[k] << endl;
    return 0;
}
```

a. Prototype for read:

b. Prototype for reorder:

c. Prototype for indexOfNextContain:

5. [5pts] What problem would arise if a program were to call the function foo defined below over and over and over again? Note: What value this function computes is not really relevant to answering this problem.

```cpp
double foo(double x, double d, int n)
{
    double* D = new double[n+1];
    for(int i = 0; i <= n; i++)
        D[i] = x + i*d;
    while(n-- > 1)
        for(int i = 0; i < n; i++)
            D[i] = D[i+1] - D[i];
    return D[0];
}
```
6. [10pts]

a. When I run the program below, no matter what the user types, it crashes:

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

void specialPrint(int* A, int N, char c);

int main()
{
    char c = ',',
    int* A;
    int N = 10;
    for(int i = 0; i < N; i++)
        cin >> A[i];
    cout << "Made it here!" << endl;
    specialPrint(A,N,c);
    return 0;
}
```

b. When the Part a bug is fixed, the program no longer crashes, but it doesn't do what it's supposed to do, which is print the inputted 10 numbers in reverse order separated by commas. Instead here's what we get:

```
~/ $ ./ex03
100 90 80 70 60 50 40 30 20 10
Made it here!
135121,10,20,30,40,50,60,70,80,100
```

Show how to fix specialPrint!
7. [9 pts] When I try to compile the code below, I get the following error messages:

```
ex.cpp:39:26: error: invalid initialization of reference of type 'char&' from expression of type 'std::string'
ex.cpp:14:30: error: cannot convert 'std::string*' to 'std::string*' in initialization
ex.cpp:18:6: error: initializing argument 3 of 'void show(std::string*, int, std::ostream')
```

Annotate the code to show how to fix these errors.

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

void show(string *A, int n, ostream out);
void modify(char& c, char bad, char cover);
void cross(string *A, int n, char bad);

int main()
{
    int num;
    cin >> num;
    string* B = new string*[num];
    for(int i = 0; i < num; i++)
        cin >> B[i];
    cross(B,num,'o');
    show(B,num,cout);
    return 0;
}

void show(string *A, int n, ostream out)
{
    for(int i = 0; i < n; i++)
        out << A[i] << ' ';
    out << endl;
}

void modify(char& c, char bad, char cover)
{
    if (c == bad)
        c = cover;
}

void cross(string *A, int n, char bad)
{
    for(int r = 0; r < n; r++)
        for(int i = 0; i < A[r].length(); i++)
            modify(A[i],bad,'x');
}
```

8. [8 pts] I'd like a function "convert" that converts its argument from a time in seconds to a time in minutes, and returns the leftover time (in seconds). For example, this code

```cpp
int t = 137;
double left = convert(t);
cout << t << " minutes and " << left << " seconds";
```

... should print out "2 minutes and 17 seconds". Define the function "convert" that works this way.
9. [15pts] Consider the following function prototypes and variable declarations (assume arrays get allocated and initialized elsewhere).

```cpp
int rev(int k); // returns the reverse of k, e.g. if k is 387, return 783
string rev(string s); // returns the reverse of s
void capit(char &c); // changes c to a capital letter if it's lower case
bool iscap(char c); // returns true if c is a capital letter, false otherwise
string capit(string s); // returns the capitalized version of s
```

```cpp
c char c = 'a';
string w = "hello";
char *A;
string *B; // assume arrays get allocated and initialized elsewhere!
int** C;
```

For each of the following, one expression is OK, the other one is not. Circle the expression that IS NOT OK and explain why it's not OK but the other one is.

a. `rev(B[0])` \[ \text{rev(B)} \]

b. `capit(B[1][2])` \[ \text{capit(c + 1)} \]

c. `cout << capit(w)` \[ `cout << capit(c)` \]

d. `rev("four")[0]` \[ `rev(4)[0]` \]

e. `A[0][1]` \[ `C[0][1]` \]