QUEENS COLLEGE Department of Computer Science
CSCI 111 Second Midterm Exam Spring 2017 05.10.17

Solutions

09.00am – 09.50am, Wednesday, May 10, 2017

Problem 1 *(points)* Write the best **title lines** for the functions that are called by the following main program. **Do not supply blocks for the functions.**

```
int main() {
   char x = 'a', y = 'b', z = 'c';
   string a[3] = {"A", "B", "Freddy"};
   bool b[2][2] = {{true, false},{true,true}};
   int c = 0;

   c = subtract(z, y);                  // (a) sets c to the difference 1
   welcomeUser(a[2]);                   // (b) print out "Hello Freddy"
   deFred(a[2]);                        // (c) change it to "Anon"
   reset(b, 2, 2, 2 == 2);              // (d) sets the array to be all true
   cout << addOn(addOn(a[2],x),y);      // (e) function adds on a character
   return 0;
}
```

(a) Title line for **subtract**.
Answer:

```
int subtract(char a, char b)
```

(b) Title line for **welcomeUser**.
Answer:

```
void welcomeUser(string s)
```

(c) Title line for **deFred**.
Answer:

```
void deFred(string &s)
```

(d) Title line for **reset**.
Answer:

```
void reset(bool b[][2], int r, int c, bool x)
```

(e) Title line for **addOn**.
Answer:

```
string addOn(string s, char c)
```

Problem 2 *(points)* Consider the following C++ program.

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

void up(int x[][3], int rows, int cols) {
   for (int c = 0; c < cols; c++) for (int r = 0; r < rows; r++)
      cout << (char) ('A' + x[r][c]);
   cout << endl;
}

void recursive(int x[][3], int r) {
    if (r == 0) {
       cout << endl;
       return;
    }
    cout << x[r - 1][r - 1];
    recursive(x, r - 1);
}

int main() {
    int x[3][3] = {{3, 1, 4}, {1, 5, 9}, {2, 6, 5}};
    cout << x[1][1] << x[0][2] << endl;                  // line (a)
    cout << x[x[1][0]][x[1][0]] << endl;                 // line (b)
    for (int c = 0; c < 3; c++) cout << x[2][c] << endl; // line (c)
    up(x, 2, 2);                                         // line (d)
    recursive(x,3);                                      // line (e)
}
```

(a) What is the output at line (a)?
Answer:

```
54
```

(b) What is the output at line (b)?
Answer:

```
5
```

(c) What is the output at line (c)?
Answer:

```
2
6
5
```

(d) What is the output at line (d)?
Answer:

```
DBBF
```

(e) What is the output at line (e)?
Answer:

```
553
```

Problem 3 *(points)* Write a function called *goodStudent* that gives the name of a student who scores at least 8 points on a quiz. The function uses three parameters: an array of names, an array of scores and a count of students. If more than one student scores at least 8, the first name in the array with a score of at least 8 is returned. If no student does well a result of "Nobody" is returned.

For example, a program that uses the function *goodStudent* follows.

```
int main() {
   string students[4] = {"Freddy", "Kelly", "Arthur", "Jack"};
   int scores[4] = {0, 8, 7, 10};
   int hardQuiz[4] = {0, 1, 1, 2};
   cout << goodStudent(students, scores, 4) << endl;   // prints Kelly
   cout << goodStudent(students, hardQuiz, 4) << endl; // prints Nobody
   return 0;
}
```

Answer:

```
string goodStudent(string names[], int scores[], int count) {
   for (int c = 0; c < count; c++)
     if (scores[c] >= 8) return names[c];
   return "Nobody";
}
```

Problem 4 *(points)* Write a function called *biggerDigits* that uses two positive integer parameters with the same number of digits and returns an integer whose digit in each position is the bigger of the two digits in that position in the input parameters. If a negative parameter is given, or if parameters with unequal numbers of digits are given your function can return any result of your choosing.

For example, a program that uses the function *biggerDigits* follows.

```
int main() {
   cout << biggerDigits(567, 765) << endl;          // prints 767
   cout << biggerDigits(123456, 444444) << endl;    // prints 444456
   cout << biggerDigits(999, 111) << endl;          // prints 999
   return 0;
}
```

Answer:

```
int biggerDigits(int a, int b) {
   if (a == 0 && b == 0) return 0;
   if (a % 10 > b % 10) return 10 * biggerDigits(a/10, b/10) + a % 10;
   else return 10 * biggerDigits(a/10, b/10) + b % 10;
}
```

QUEENS COLLEGE Department of Computer Science
CSCI 111 Second Midterm Exam Spring 2017 05.10.17

Solutions

02.45pm – 03.35pm, Wednesday, May 10, 2017

Problem 1 *(points)* Write the best **title lines** for the functions that are called by the following main program. **Do not supply blocks for the functions.**

```
int main() {
   string name = "Freddy Next Door";
   int a2[2][3] = {{-2, 4, 3}, {-3, 4, 2}};
   int a[5] = {7, 6, 5, 9, 7};
   cout << firstLetters(name, name) << endl;      // (a) prints: F F
   cout << sumAll(a, 5, a, 5) << endl;            // (b) prints: 68 by summing twice
   cout << middleInitial(name) << endl;           // (c) prints: N
   makeRandom(a2, 2, 3);                          // (d) reset the array with random entries
   if (countIt(name, countIt(middleInitial(name), 5.0)) > 0) // (e) mystery
      cout << "Yes\n";
   return 0;
}
```

(a) Title line for **firstLetters**.
Answer:

```
string firstLetters(string x, string y)
```

(b) Title line for **sumAll**.
Answer:

```
int sumAll(int x[], int c, int y[], int d)
```

(c) Title line for **middleInitial**.
Answer:

```
string middleInitial(string x)
```

(d) Title line for **makeRandom**.
Answer:

```
void makeRandom(int x[][3], int r, int c)
```

(e) Title line for **countIt**.
Answer:

```
double countIt(string s, double x)
```

Problem 2 *(points)* Write blocks of code to perform the functions used in the following main program. Your blocks must match the given title lines. Each block should be a short function of only a few lines.

```
int main() {
   int b = 1, c = 2, a[4] = {3, 1, 4, 1}, x = 10, y = 1000;
 // (a) Finds the cube, here -27
   cout << cube(-3) << endl;
 // (b) Finds a random number between 1 and x
   cout << random(x) << endl;
 // (c) Prints the ratio as a percentage, here 12.5% for 1/8
   cout << percentage(1, 8) << "%" << endl;
 // (d) reverse print the array here 1413  (no spaces)
   reversePrint(a, 4);
 // (e) determine whether x or y has more digits, assume x and y both positive
   if (hasMore(x,y)) cout << "x is longer\n";
   return 0;
}
```

Answer:

(a)

```
int cube(int x) {
    return x * x * x;
}
```

(b)

```
int random(int x) {
   return rand() % x + 1;
}
```

(c)

```
double percentage(int x, int y) {
    return 100.0 * x / y;
}
```

(d)

```
void reversePrint(int x[], int cap) {
   for (int i = cap - 1; i >= 0; i--)
       cout << x[i];
   cout << endl;
}
```

(e)

```
bool hasMore(int x, int y) {
   if (x < 10) return false;
   if (y < 10) return true;
   return hasMore(x / 10, y / 10);
}
```

Problem 3 *(points)* Consider the following C++ program.

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

int xy(int x, string &y) {
  if (x > 0) y = "error";
  else y = "fine";
  if (x <= 0) return 3;
  return x % 10 + 10 * xy(x/10, y);
}

int main() {
    int c = 9, x = 10;
    string y;
    if ((x % c) >= (c % x)) cout << c;        // line (a)
    cout << endl;
    for(c = 8; c > x - c; c--) cout << c;     // line (b)
    cout << endl;
    cout << xy(-2, y) << endl;                // line (c)
    cout << y << endl;                        // line (d)
    cout << xy(3145, y) << endl;              // line (e)
}
```

(a) What is the output at line (a)?
Answer:

(b) What is the output at line (b)?
Answer:

```
876
```

(c) What is the output at line (c)?
Answer:

```
3
```

(d) What is the output at line (d)?
Answer:

```
fine
```

(e) What is the output at line (e)?
Answer:

```
33145
```

Problem 4 *(points)* Write a function called *toNumber* that uses an array of integers each entry of which is between 0 and 9 and returns an integer formed by using the entries as its digits. If input array entries are out of range, you can return any answer of your choosing. Your function should not use more than 5 lines of code.

For example, a program that uses the function *toNumber* follows.

```
int main() {
   int a[6] = {3,1,4,1,5,9};
   cout << toNumber(a, 6) << endl;          // prints 314159
   cout << toNumber(a, 6) + 1 << endl;      // prints 314160
   return 0;
}
```

Answer:

```
int toNumber(int x[], int c) {
   if (c <= 1) return x[0];
   return 10 * toNumber(x, c - 1) + x[c - 1];
}
```