QUEENS COLLEGE Department of Computer Science
CSCI 111 Final Exam Spring 2018 05.21.18

Solutions

08.30am – 10.30am, Monday, May 21, 2018

Problem 1 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() {
   int z[3] = {0, 1, 2};
   double r[3] = {1.9, 2.3, 3.0};
   bool bl = true;

   bl = a(bl, bl);                // (a)
   r[0] = b(z[0], r[1]);          // (b)
   z[2] = c(a(bl,bl));            // (c)
   d(d(z[0], z[1]), 5);           // (d)
   a(e(z[0] + z[1], r, z), bl);   // (e)
   return 0;
}
```

(a) Title line for **a**.

Answer:

```
bool a(bool x, bool y)
```

(b) Title line for **b**.

Answer:

```
double b(int x, double u)
```

(c) Title line for **c**.

Answer:

```
int c(bool x)
```

(d) Title line for **d**.

Answer:

```
int d(int x, int y)
```

(e) Title line for **e**.

Answer:

```
bool e(int x, double y[], int z[])
```

Problem 2 Consider the following C++ program. It is compiled to **a.out** and executed with the command **./a.out abcabc abc123**.

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

int main(int argc, char *argv[]) {
    string words[4] = {"A ", "very ", "easy", "question "};
    cout << words[1].substr(2) << endl;                                    // line (a)
    for (int i = 0; i <= 2; i++) cout << words[i][i];  cout << endl;   // line (b)
    cout << argv[1] << endl;                                               // line (c)
    cout << words[3].find("u") << endl;                                    // line (d)
    cout << argc << endl;                                                  // line (e)
    return 0;
}
```

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

Answer:

```
ry
```

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

Answer:

```
Aes
```

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

Answer:

```
abcabc
```

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

Answer:

```
1
```

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

Answer:

```
3
```

Problem 3 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 that is written only in the designated answer space.

```
int main() {
   int x[5] = {3, 1, 4, 1, 5};
// (a) Return the average.  Here 2.5 is printed.
   cout << average(2, 3) << endl;
// (b) Return the middle of 3 numbers, here 5 is printed.
   cout << middle(5, 6, 4) << endl;
// (c) Return the middle entry of an array with odd capacity.  Here 4.
   cout << middleEntry(x, 5) << endl;
// (d) Return the first index of 7 in the array or -1 if not present. Here -1 is printed.
   cout << findIndex7(x, 5) << endl;
// (e) Return the upper case version of a lower case char. Here print H
   cout << toUpper('h') << endl;
   return 0;
}
```

Answer:

(a)

```
double average(int x, int y) {
    return (x + y) / 2.0;
}
```

(b)

```
int middle(int x, int y, int z) {
   if ((x - y) * (x - z) <= 0) return x;
   if ((y - x) * (y - z) <= 0) return y;
   return z;
}
```

(c)

```
int middleEntry(int x[], int c) {
   return x[(c - 1)/2];
}
```

(d)

```
int findIndex7(int array[], int cap) {
   for (int i = 0; i < cap; i++)
      if (array[i] == 7) return i;
   return -1;
}
```

(e)

```
char toUpper(char x) {
   return x + 'A' - 'a';
}
```

Problem 4 Write a complete C++ program that does the following:

It generates 250 random numbers between 1 and 1000. For each of the 250 numbers that has not been seen before it prints the number.

Your program should not repeat any output value, random values should be computed with the C++ random number function `rand()` . This function should be called exactly 250 times. It is likely that fewer than 250 numbers will be printed.

Excessively long or complicated code may lose points.

Answer:

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

int main() {
   bool seen[1001];
   for (int i = 0; i < 1001; i++) seen[i] = false;
   for (int i = 0; i < 250; i++) {
      int n = rand() % 1000 + 1;
      if (!seen[n]) {
         cout << n << endl;
         seen[n] = true;
      }
   }
   return 0;
}
```

Problem 5 Write a function called `rowSums`. The function has two array parameters `first` and `second` the first is two dimensional with 5 columns and the second is one dimensional. The entries of both arrays have type `int`. Additional parameters specify the row and column counts for `first` . The function sets each entry `second[r]` to be the sum of the entries in row `r` of `first` .

For example, a program that uses the function follows.

```
int main() {
   int first[3][5] = {{9,9,8,1,0},{2,9,8,1,0},{1,1,8,1,0}};
   int second[3];
   rowSums(first, second, 3, 5);
   for (int i = 0; i < 3; i++) cout << second[i] << " ";  // prints 27 20 12
   cout << endl;
   return 0;
}
```

Excessively long or complicated code may lose points.

Answer:

```
void rowSums(int first[][5], int second[], int r, int c) {
    for (int i = 0; i < r; i++) {
       second[i] = 0;
       for (int j = 0; j < c; j++)
           second[i] += first[i][j];
    }
}
```

Problem 6 Write a function called `reverseAdd`. The function has two integer parameters `first` and `second` that are positive. It returns the number obtained by attaching the reverse of `second` after `first`. For instance `reverseAdd(27,729)` would return 27927, If parameters have illegal values your function can operate however you choose. Excessively long solutions that use more than 8 lines of code may lose points.

For example, a program that uses the function follows.

```
int main() {
   cout << reverseAdd(16, 538) << endl;     // prints 16835
   cout << reverseAdd(862, 538) << endl;    // prints 862835
   return 0;
}
```

Answer:

```
int reverseAdd(int first, int second) {
   if (second == 0) return first;
   return reverseAdd(first * 10 + second % 10, second / 10);
}
```

QUEENS COLLEGE Department of Computer Science
CSCI 111 Final Exam Spring 2018 05.21.18

Solutions

08.30am – 10.30am, Monday, May 21, 2018

Problem 1 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() {
   double dd[3] = {0, 1.1, 2.2};
   string st[3] = {"1.9", "2.3", "3.0"};

   dd[1] = f1(dd[2] + dd[1], dd[2]);                  // (a)
   st[0] = f2(dd[0] + dd[1], dd[0], dd[0], st[2]);    // (b)
   dd[1] = f3(st, st, 3);                             // (c)
   f4(st[1], 1);                                      // (d)
   char k = f4(f5(dd[1], st), dd[1]);                 // (e)
   return 0;
}
```

(a) Title line for **f1**.

Answer:

```
double f1(double x, double y)
```

(b) Title line for **f2**.

Answer:

```
string f2(double x, double y, double z, string w)
```

(c) Title line for **f3**.

Answer:

```
double f3(string w[], string x[], int c)
```

(d) Title line for **f4**.

Answer:

```
char f4(string x, double y)
```

(e) Title line for **f5**.

Answer:

```
string f5(double x, string y[])
```

Problem 2 Consider the following C++ program. It is compiled to **a.out** and executed with the command **./a.out 123cba abcxyz ABCDEF**.

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

int main(int argc, char *argv[]) {
    string words[4] = {"Max", "Freddy", "Jack", "Kelly"};
    cout << words[1].substr(2) << endl;                                    // line (a)
    for (int i = 0; i <= 2; i++) cout << words[i][i];  cout << endl;   // line (b)
    cout <<  argv[2] << endl;                                              // line (c)
    cout << words[3].rfind("l") << endl;                                   // line (d)
    cout << argc << endl;                                                  // line (e)
    return 0;
}
```

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

Answer:

```
eddy
```

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

Answer:

```
Mrc
```

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

Answer:

```
abcxyz
```

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

Answer:

```
3
```

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

Answer:

```
4
```

Problem 3 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 that is written only in the designated answer space.

```
int main() {
   int x[5] = {7, 1, 4, 7, 5};
// (a) Return the average.  Here 3.33333 is printed.
   cout << average(2, 3, 5) << endl;
// (b) Return the smaller of 2 numbers, here 5 is printed.
   cout << smaller(5, 6) << endl;
// (c) Return the next to last entry of an array.  Here 7.
   cout << secondLastEntry(x, 5) << endl;
// (d) Return the last index of 7 in the array or -1 if not present. Here 3 is printed.
   cout << findIndex7(x, 5) << endl;
// (e) Return the lower case version of an upper case char. Here print h
   cout << toLower('H') << endl;
   return 0;
}
```

Answer:

(a)

```
double average(int x, int y, int z) {
    return (x + y + z) / 3.0;
}
```

(b)

```
int smaller(int x, int y) {
   if (x <= y) return x;
   return y;
}
```

(c)

```
int secondLastEntry(int x[], int c) {
   return x[c - 2];
}
```

(d)

```
int findIndex7(int array[], int cap) {
   for (int i = cap - 1; i >= 0; i--)
      if (array[i] == 7) return i;
   return -1;
}
```

(e)

```
char toLower(char x) {
   return x + 'a' - 'A';
}
```

Problem 4 Write a complete C++ program that does the following:

It generates 250 random numbers between 1 and 1000. For each of the 250 numbers that has been seen before it prints the number.

Random values should be computed with the C++ random number function `rand()` . This function should be called exactly 250 times. If a random number is seen more than twice, it will be printed more than once.

Excessively long or complicated code may lose points.

Answer:

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

int main() {
   bool seen[1001];
   for (int i = 0; i < 1001; i++) seen[i] = false;
   for (int i = 0; i < 250; i++) {
      int n = rand() % 1000 + 1;
      if (seen[n]) cout << n << endl;
      seen[n] = true;
   }
   return 0;
}
```

Problem 5 Write a function called `colSums`. The function has two array parameters `first` and `second` the first is two dimensional with 5 columns and the second is one dimensional with the same number of columns. The entries of both arrays have type `int`. Additional parameters specify the row and column counts for `first` . The function sets each entry `second[c]` to be the sum of the entries in column `c` of `first` .

For example, a program that uses the function follows.

```
int main() {
   int first[3][5] = {{9,9,8,1,0},{2,9,8,1,0},{1,1,8,1,0}};
   int second[5];
   colSums(first, second, 3, 5);
   for (int i = 0; i < 5; i++) cout << second[i] << " ";  // prints 12 19 24 3 0
   cout << endl;
   return 0;
}
```

Excessively long or complicated code may lose points.

Answer:

```
void colSums(int first[][5], int second[], int r, int c) {
    for (int i = 0; i < c; i++) {
       second[i] = 0;
       for (int j = 0; j < r; j++)
           second[i] += first[j][i];
    }
}
```

Problem 6 Write a function called `attach`. The function has two integer parameters `first` and `second` that are positive. It returns the number obtained by attaching `second` after `first` . For instance `attach(27,927)` would return 27927, If parameters have illegal values your function can operate however you choose. Excessively long solutions that use more than 8 lines of code may lose points.

For example, a program that uses the function follows.

```
int main() {
   cout << attach(16, 835) << endl;     // prints 16835
   cout << attach(862, 835) << endl;    // prints 862835
   return 0;
}
```

Answer:

```
int attach(int first, int second) {
   if (second == 0) return first;
   return attach(first, second / 10) * 10 + second % 10;
}
```

QUEENS COLLEGE Department of Computer Science
CSCI 111 Final Exam Spring 2018 05.21.18

Solutions

01.45pm – 03.45pm, Monday, May 21, 2018

Problem 1 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() {
   int dd[3] = {0, 1, 2};
   char st[3] = {'9', '2', '3'};

   dd[1] = f1(dd[2] + dd[1], dd[2]);                  // (a)
   st[0] = f2(dd[0] + dd[1], dd[0], dd[0], st[2]);    // (b)
   if (f3(st, st, 3)) return 0;                       // (c)
   cout << 2 + f4(st[1], 1);                          // (d)
   f4(f5(dd[1], st), dd[1]);                          // (e)
   return 0;
}
```

(a) Title line for **f1**.

Answer:

```
int f1(int x, int y)
```

(b) Title line for **f2**.

Answer:

```
char f2(int x, int y, int z, char w)
```

(c) Title line for **f3**.

Answer:

```
bool f3(char w[], char x[], int c)
```

(d) Title line for **f4**.

Answer:

```
int f4(char x, int y)
```

(e) Title line for **f5**.

Answer:

```
char f5(int x, char y[])
```

Problem 2 Consider the following C++ program. It is compiled to **a.out** and executed with the command **./a.out 123456 456789 135799**.

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

int main(int argc, char *argv[]) {
    string words[4] = {"123", "987654", "9999", "77777"};
    cout << words[2].substr(2) << endl;                                    // line (a)
    for (int i = 1; i <= 3; i++) cout << words[i][i];  cout << endl;   // line (b)
    words[3] = argv[2];
    cout << words[3] << endl;                                              // line (c)
    cout << words[3].find("9") << endl;                                    // line (d)
    cout << argc << endl;                                                  // line (e)
    return 0;
}
```

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

Answer:

```
99
```

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

Answer:

```
897
```

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

Answer:

```
456789
```

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

Answer:

```
5
```

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

Answer:

```
4
```

Problem 3 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 that is written only in the designated answer space.

```
int main() {
   int x[5] = {7, 1, 4, 7, 5};
// (a) Return the average.  Here 3.33333 is printed.
   cout << average(2, 3, 5) << endl;
// (b) Return the smaller of 2 numbers, here 5 is printed.
   cout << smaller(5, 6) << endl;
// (c) Return the next to last entry of an array.  Here 7.
   cout << secondLastEntry(x, 5) << endl;
// (d) Return the last index of 7 in the array or -1 if not present. Here 3 is printed.
   cout << findIndex7(x, 5) << endl;
// (e) Return the lower case version of an upper case char. Here print h
   cout << toLower('H') << endl;
   return 0;
}
```

Answer:

(a)

```
double average(int x, int y, int z) {
    return (x + y + z) / 3.0;
}
```

(b)

```
int smaller(int x, int y) {
   if (x <= y) return x;
   return y;
}
```

(c)

```
int secondLastEntry(int x[], int c) {
   return x[c - 2];
}
```

(d)

```
int findIndex7(int array[], int cap) {
   for (int i = cap - 1; i >= 0; i--)
      if (array[i] == 7) return i;
   return -1;
}
```

(e)

```
char toLower(char x) {
   return x + 'a' - 'A';
}
```

Problem 4 Write a complete C++ program that does the following:

It generates random numbers between 1 and 1000. As soon as it generates a number that is the sum of the two numbers right before it, it should print out this sum, report the total number of random numbers that have been generated and end.

Your program should produce output in this form:

```
626 = 154 + 472
Generated 307 Numbers
```

Random values should be computed with the C++ random number function `rand()` .

Excessively long or complicated code may lose points.

Answer:

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

int main() {
   int lastButOne = rand() % 1000 + 1;
   int last = rand() % 1000 + 1;
   int count = 2;
   while (true) {
      int n = rand() % 1000 + 1;
      count++;
      if (n == last + lastButOne) {
         cout << n << " = " << last << " + " << lastButOne << endl;
         cout << "Generated " << count << " Numbers " << endl;
         return 0;
      }
      lastButOne = last;
      last = n;
   }
   return 0;
}
```

Problem 5 Write a function called `rowPositive`. The function has two array parameters `first` and `second` the first is two dimensional with 5 columns and the second is one dimensional. The entries of `first` have type `int`. Additional parameters specify the row and column counts for `first` . The function sets each entry `second[r]` to be `true` exactly when the entries in row `r` of `first` have a positive sum.

For example, a program that uses the function follows.

```
int main() {
   int first[3][5] = {{9,-9,8,1,0},{2,-9,-8,1,0},{1,-1,-8,1,0}};
   bool second[3];
   rowPositive(first, second, 3, 5);
   for (int i = 0; i < 3; i++)
      if (second[i]) cout << "Positive ";
      else cout << "Negative ";  // prints Positive Negative Negative
   cout << endl;
   return 0;
}
```

Excessively long or complicated code may lose points.

Answer:

```
void rowPositive(int first[][5], bool second[], int r, int c) {
    for (int i = 0; i < r; i++) {
       int sum = 0;
       for (int j = 0; j < c; j++)
           sum += first[i][j];
       second[i] = sum > 0;
    }
}
```

Problem 6 Write a function called `numberMatch`. The function has two integer parameters `first` and `second` that are positive and have the same number of digits. It returns the number of positions where their digits are equal. For instance `numberMatch(1234,1894)` would return 2, since the numbers match in their first and last digits. If parameters have illegal values your function can operate however you choose. Excessively long solutions that use more than 5 lines of code may lose points.

For example, a program that uses the function follows.

```
int main() {
   cout << numberMatch(1628, 1328) << endl;      // prints 3
   cout << numberMatch(862, 862)  << endl;       // prints 3
   cout << numberMatch(862, 628)  << endl;       // prints 0
   return 0;
}
```

Answer:

```
int numberMatch(int first, int second) {
   if (second == 0) return 0;
   int ans = numberMatch(first / 10, second / 10);
   if (first % 10 == second % 10) return ans + 1;
   return ans;
}
```

QUEENS COLLEGE Department of Computer Science
CSCI 111 Final Exam Spring 2018 05.21.18

Solutions

01.45pm – 03.45pm, Monday, May 21, 2018

Problem 1 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() {
   double rr[3] = {0, 1.1, 2.2};
   int zz[3] = {19, 23, 30};
   char ch = 'a';

   ch = a(ch, ch);                    // (a)
   zz[0] = b(rr[0], zz[1]);           // (b)
   cout << 3 * c(a(ch,ch));           // (c)
   d(d(rr[0], rr[1]), 5);             // (d)
   a(e(rr, rr[0] + rr[1], zz), ch);   // (e)
   return 0;
}
```

(a) Title line for **a**.

Answer:

```
char a(char x, char y)
```

(b) Title line for **b**.

Answer:

```
int b(double x, int u)
```

(c) Title line for **c**.

Answer:

```
int c(char x)
```

(d) Title line for **d**.

Answer:

```
double d(double x, double y)
```

(e) Title line for **e**.

Answer:

```
char e(double z[], double x, int y[])
```

Problem 2 Consider the following C++ program. It is compiled to **a.out** and executed with the command **./a.out PQRPQR PQR789**.

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

int main(int argc, char *argv[]) {
    string words[4] = {"NOP", "NOPQ", "OPQR", "MNOPQRS"};
    cout << words[3].substr(2) << endl;                                    // line (a)
    for (int i = 0; i <= 3; i++) cout << words[i][i];  cout << endl;   // line (b)
    words[3] = argv[2];
    cout << words[3] << endl;                                              // line (c)
    cout << words[3].rfind("R") << endl;                                   // line (d)
    cout << argc % argc << endl;                                           // line (e)
    return 0;
}
```

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

Answer:

```
OPQRS
```

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

Answer:

```
NOQP
```

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

Answer:

```
PQR789
```

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

Answer:

```
2
```

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

Answer:

```
0
```

Problem 3 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 that is written only in the designated answer space.

```
int main() {
   int x[5] = {3, 1, 4, 1, 5};
// (a) Return the average.  Here 2.5 is printed.
   cout << average(2, 3) << endl;
// (b) Return the middle of 3 numbers, here 5 is printed.
   cout << middle(5, 6, 4) << endl;
// (c) Return the middle entry of an array with odd capacity.  Here 4.
   cout << middleEntry(x, 5) << endl;
// (d) Return the first index of 7 in the array or -1 if not present. Here -1 is printed.
   cout << findIndex7(x, 5) << endl;
// (e) Return the upper case version of a lower case char. Here print H
   cout << toUpper('h') << endl;
   return 0;
}
```

Answer:

(a)

```
double average(int x, int y) {
    return (x + y) / 2.0;
}
```

(b)

```
int middle(int x, int y, int z) {
   if ((x - y) * (x - z) <= 0) return x;
   if ((y - x) * (y - z) <= 0) return y;
   return z;
}
```

(c)

```
int middleEntry(int x[], int c) {
   return x[(c - 1)/2];
}
```

(d)

```
int findIndex7(int array[], int cap) {
   for (int i = 0; i < cap; i++)
      if (array[i] == 7) return i;
   return -1;
}
```

(e)

```
char toUpper(char x) {
   return x + 'A' - 'a';
}
```

Problem 4 Write a complete C++ program that does the following:

It generates random numbers between 1 and 1000. As soon as it generates a number that is the square root of the number right before it, it should print out these two numbers, report the total number of random numbers that have been generated and end.

Your program should produce output in this form:

```
19 = sqrt 361
Generated 20703 Numbers
```

Random values should be computed with the C++ random number function `rand()` .

Excessively long or complicated code may lose points.

Answer:

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

int main() {
   int last = rand() % 1000 + 1;
   int count = 1;
   while (true) {
      int n = rand() % 1000 + 1;
      count++;
      if (n * n == last) {
         cout << n << " = sqrt " << last << endl;
         cout << "Generated " << count << " Numbers " << endl;
         return 0;
      }
      last = n;
   }
   return 0;
}
```

Problem 5 Write a function called `colPositive`. The function has two array parameters `first` and `second` the first is two dimensional with 5 columns and the second is one dimensional with the same number of columns. The entries of `first` have type `int`. Additional parameters specify the row and column counts for `first` . The function sets each entry `second[c]` to be `true` exactly when the entries in column `c` of `first` have a positive product.

For example, a program that uses the function follows.

```
int main() {
   int first[3][5] = {{9,-9,8,1,0},{2,-9,-8,1,0},{1,-1,-8,1,0}};
   bool second[5];
   colPositive(first, second, 3, 5);
   for (int i = 0; i < 5; i++)
      if (second[i]) cout << "Positive ";
      else cout << "Not ";  // prints Positive Not Positive Positive Not
   cout << endl;
   return 0;
}
```

Excessively long or complicated code may lose points.

Answer:

```
void colPositive(int first[][5], bool second[], int r, int c) {
    for (int i = 0; i < c; i++) {
       int product = 1;
       for (int j = 0; j < r; j++)
           product *= first[j][i];
       second[i] = product > 0;
    }
}
```

Problem 6 Write a function called `sumMatch`. The function has two integer parameters `first` and `second` that are positive and have the same number of digits. It returns the sum of the digits that match in the two numbers. For instance `sumMatch(1254,1451)` would return 6, since the numbers match in their first and third digits which are 1 and 5 the answer is found as $1 + 5$. If parameters have illegal values your function can operate however you choose. Excessively long solutions that use more than 5 lines of code may lose points.

For example, a program that uses the function follows.

```
int main() {
   cout << sumMatch(1628, 1328) << endl;      // prints 11
   cout << sumMatch(862, 862)  << endl;       // prints 16
   cout << sumMatch(862, 628)  << endl;       // prints 0
   return 0;
}
```

Answer:

```
int sumMatch(int first, int second) {
   if (second == 0) return 0;
   int ans = sumMatch(first / 10, second / 10);
   if (first % 10 == second % 10) return ans + first % 10;
   return ans;
}
```

QUEENS COLLEGE Department of Computer Science
CSCI 111 Final Exam Spring 2018 05.17.18

Solutions

07.00am – 09.00am, Thursday, May 17, 2018

Problem 1 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() {
   double dd[3] = {0, 1.1, 2.2};
   string st[3] = {"1.9", "2.3", "3.0"};

   dd[1] = f1(dd[2] + dd[1], dd[2]);                  // (a)
   st[0] = f2(dd[0] + dd[1], dd[0], dd[0], st[2]);    // (b)
   st[0][0] = f3(st, st, 3);                          // (c)
   f4(st[1], 1);                                      // (d)
   f4(f5(dd[1], st), f4("hello", dd[1]));             // (e)
   return 0;
}
```

(a) Title line for **f1**.

Answer:

```
double f1(double x, double y)
```

(b) Title line for **f2**.

Answer:

```
string f2(double x, double y, double z, string w)
```

(c) Title line for **f3**.

Answer:

```
char f3(string w[], string x[], int c)
```

(d) Title line for **f4**.

Answer:

```
double f4(string x, double y)
```

(e) Title line for **f5**.

Answer:

```
string f5(double x, string y[])
```

Problem 2 Consider the following C++ program. It is compiled to **a.out** and executed with the command **./a.out 111999 229992 999333**.

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

int main(int argc, char *argv[]) {
    string words[4] = {"444", "555555", "6666", "77777"};
    cout << words[2].substr(2) << endl;                                    // line (a)
    for (int i = 1; i <= 3; i++) cout << words[i][i];  cout << endl;   // line (b)
    words[3] = argv[2];
    cout << words[3] << endl;                                              // line (c)
    cout << words[3].rfind("9") << endl;                                   // line (d)
    cout << argc << endl;                                                  // line (e)
    return 0;
}
```

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

Answer:

66

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

Answer:

567

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

Answer:

229992

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

Answer:

4

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

Answer:

4

Problem 3 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 that is written only in the designated answer space.

```
int main() {
   string x[5] = {"CS", "111", "Queens", "College", "CUNY"};
// (a) Return the average.  Here 2.5 is printed.
   cout << average(2, 3) << endl;
// (b) Return the middle of 3 numbers, here 5 is printed.
   cout << middle(5, 6, 4) << endl;
// (c) Return the string formed by the first characters of the entries.  Here C1QCC.
   cout << initialLetters(x, 5) << endl;
// (d) Return the first index of an entry containing a target or -1 if not present. Here 2 is printed.
   cout << findIndexContains(x, 5, "ee") << endl;
// (e) Return the longest entry. Here print  College
   cout << longest(x, 5) << endl;
   return 0;
}
```

Answer:

(a)

```
double average(int x, int y) {
    return (x + y) / 2.0;
}
```

(b)

```
int middle(int x, int y, int z) {
   if ((x - y) * (x - z) <= 0) return x;
   if ((y - x) * (y - z) <= 0) return y;
   return z;
}
```

(c)

```
string initialLetters(string x[], int c) {
   string ans = "";
   for (int i = 0; i < c; i++) ans += x[i].substr(0, 1);
   return ans;
}
```

(d)

```
int findIndexContains(string array[], int cap, string target) {
   for (int i = 0; i < cap; i++) {
      int x = array[i].find(target);
      if (0 <= x && x < array[i].size()) return i;
   }
   return -1;
}
```

(e)

```
string longest(string x[], int cap) {
   string ans = x[0];
```

```
   for (int i = 1; i < cap; i++)
      if (x[i].size() > ans.size()) ans = x[i];
   return ans;
}
```

Problem 4 Write a complete C++ program that does the following:

It generates random numbers between 1 and 1000. As soon as a repeat number is generated the program stops and reports the total number of random numbers that have been generated.

Random values should be computed with the C++ random number function `rand()` .

Excessively long or complicated code may lose points.

Answer:

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

int main() {
   int count = 0;
   bool seen[1001];
   for (int i = 0; i < 1001; i++) seen[i] = false;
   while (true) {
      int n = rand() % 1000 + 1;
      count++;
      if (seen[n]) {
         cout << count << endl;
         return 0;
      }
      seen[n] = true;
   }
   return 0;
}
```

Problem 5 Write a function called `maxIndex`. The function has two array parameters `first` and `second` the first is two dimensional with 4 columns and the second is one dimensional. The entries of the two dimensional array are required to be distinct integers. The arrays have the same number of columns. Additional parameters specify the row and column counts for `first` . The function sets each entry `second[c]` to be the index of the row of `first` for which the entry in column `c` is as large as possible.

For instance if `first` has 3 rows and 4 columns, as follows:

```
99   95   80  16
25   98   82  17
10   11   83  15
```

Then `second` would be set to store 0, 1, 2, 1.

For example, a program that uses the function follows.

```
int main() {
   int first[3][4] = {{99,95,80,16},{25,98,82,17},{10,11,83,15}};
   int second[4];
   maxIndex(first, second, 3, 4);
   for (int i = 0; i < 4; i++) cout << second[i] << " ";  // prints 0 1 2 1
   cout << endl;
   return 0;
}
```

Excessively long or complicated code may lose points.

Answer:

```
void maxIndex(int first[][4], int second[], int r, int c) {
    for (int i = 0; i < c; i++) {
       second[i] = 0;
       for (int j = 0; j < r; j++)
           if (first[j][i] > first[second[i]][i])
               second[i] = j;
    }
}
```

Problem 6 Write a function called `drop`. The function has two integer parameters `first` and `second` that are positive. It returns the number obtained by dropping digits from the left of `first` until it has no more digits than `second` . For instance `drop(19683,729)` would drop 2 digits from the left of 19683 and return 683, If parameters have illegal values your function can operate however you choose. Excessively long solutions that use more than 8 lines of code may lose points.

For example, a program that uses the function follows.

```
int main() {
   cout << drop(16, 538) << endl;     // prints 16
   cout << drop(862, 538) << endl;    // prints 862
   cout << drop(3862, 538) << endl;   // prints 862
   cout << drop(53862, 538) << endl;  // prints 862
   return 0;
}
```

Answer:

```
int drop(int first, int second) {
   if (second == 0) return 0;
   return drop(first / 10, second / 10) * 10 + first % 10;
}
```