Winter 2011 **CPE 101:** Fundamentals of Computer Science I Alexander Dekhtyar

Homework 3...

Assignment Preparation

Due date: Wednesday, March 1, beginning of the class

There is no late submission policy.

This homework is a paper-and-pencil assignment designed to help you prepare for the midterm on March 1.

Collaboration. This is an **individual assignment**, no collaboration is permitted.

Testing. All `C` code in the assignment has been tested. I suggest that you do the following, when solving the problems:

- Try solving the problem on paper. Work out the details, simulate the run of the program, converge on results.
- Type the program up, compile, debug as necessary, see it run, observe results.
- If the results do not match your answers, try to figure out where you made a mistake, and **why** the output is what it is.

Name: _____

Section: _____

Detach this section, submit it.

Problem 1. Assignment, arithmetics

Consider the following code fragment:

```
int x,y,z;

x = y/z;
y = ++x + ++z;
z = y*x;
```

For each set of initial (i.e., before the code fragment is executed) variable assignments below, specify the values of `x, y` and `z` after the code fragment executes.

(a) Initial: x: 1 y: 1 z: 1

Final: x: ___ y: ___ z: ___

(b) Initial: x: 10 y: 5 z: 10

Final: x: ___ y: ___ z: ___

(c) Initial: x: 5 y: 10 z: 5

Final: x: ___ y: ___ z: ___

Problem 2. Arrays

For each code fragment below, determine the output.

(a)

```
int x[5], i;

for (i=0;i<5;i++) {
 x[i] = i-2;
}

x[x[i]] = x[i] - x[x[i]];

printf("%d %d %d\n", x[1], x[2], x[3]);
```

(b)

```
int x[5] = {1, 3, 5, 7, 9};
int y[5],i;

for (i=0;i<5;i++) {
  if (!(x[i]%2) || !(x[i]%3)) {
      y[i] = x[i];
  }
  else {
     y[i] = x[i]*2/4;
  }
}

printf("%d %d %d %d %d\n", y[0], y[1], y[2], y[3], y[4]);
```

(c)

```
int x[5] = {5, 3, 4, 1, 2};
int i = 2;
int j;

for (j=0;j<5;j++) {
   printf("%d \n", x[i]);
   i = x[i]-1;
}
```

(d)

```
 int foo[3], bar[3];
 int k;
 for(k=1; k<=3; k++) {
   foo[k-1] = k*k;
   bar[k-1] = foo[k-1]-k/k;
  }

 printf("%d %d %d\n", foo[0], foo[1], foo[2]);
 printf("%d %d %d\n", bar[0], bar[1], bar[2]);
```

Problem 3. Simple functions

In the code fragments below all variables are declared as `int`.

(1) Consider the following code:

```
int boo(int x, int y);

int main() {
  printf("%d\n", boo(5,6));
  printf("%d\n", boo(6,5));
  printf("%d\n", boo(12,0));
  printf("%d\n", boo(-8,-17));
  return(0);
}

int boo(int x, int y){
  if (x>y) {
    return x+1;
  }
  else {
    return x+y;
  }
}
```

What will this program print?

(2) Consider the following functions:

```
 int bells(int x) {
   if (x) {
      if (!(x+1)) {
         return 0;
      }
      else {
         return x+1;
      }
   }
   return x-1;
 }

 int whistles(int x, int y) {
  if (x && y) {
      return x+y;
  }
  else {
    return x*y+1;
  }
}
```

What will the following expressions evaluate to?

(i) whistles(2,4) _____

(ii) whistles(0, 17) _____

```
(iii) bells(15)                                                    _____

 (iv) bells(bells(3))                                              _____

  (v) bells(whistles(1,2))                                         _____

 (vi) whistles(bells(1),bells(2))                                  _____

(vii) bells(whistles(bells(1),bells(3)))                           _____

 (ix) bells(bells(whistles(0,5)))                                  _____

  (x) whistles(whistles(1,3), whistles(bells(1),bells(3)))         _____
```

(c) Write a function `daysRemaining()` that takes as input two integers: one representing the calendar month (1—12) and one, representing the current date in the month. The function shall return the number of days until the first of the next month.

E.g., `daysRemaining(1,1)` returns 31, `daysRemaining(1,31)` returns 1; `daysRemaining(9,10)` returns 20, and so on.

No variable guards necessary, assume correct values for input parameters.

Problem 4. Functions with "out" parameters.

(a) Write a function `void flip()` that takes as input two integer variables and replaces the value of each of the variable with the value of the other one. Your function should use **as few** local variables as possible.

(b) Consider the following code:

```
int stub(int *x, int *y);

int main() {
   int x =1;
   int y =2;
   int z =3;

   stub(&x,&y);
   z++;
   stub(&y,&z);
   x++;
   z=stub(&z,&x);
   y++;

   printf("%d %d %d\n", x,y,z);
   return 0;
}

int stub(int *x, int *y) {
  int z;

  z = *x+*y;
  (*y)--;
  *x = *y;
  *y = *y/2;
  return z;
}
```

Show the values of variables `x,y,z` after each call to `stub`:

	x	y	z
`after stub(&x,&y);`	___	____	___
`after stub(&y,&z);`	___	____	___
`after stub(&z,&x);`	___	____	___

What will the program output?