CSE142 Exam with answers

Midterm #2
May 5, 2001
Problem numbering may differ from the test as given.

All multiple choice questions are equally weighted. You can generally assume that code shown in the questions is intended to be syntactically correct, unless something in the question or one of the answers suggests otherwise.

1. Pick the true statement.

A. "Functional decomposition" is a technique for designing a program.
B. "Functional decomposition" refers to the natural decay of a program over its lifetime.
C. A "static call graph" shows all the internal detail of the functions in a program.
D. One reason symbolic constants (#define symbols) are desirable is because they don't have to conform to the same naming rules as other identifiers.
E. Comments describing how functions work allow the compiler to double-check your function.
ANSWER: **A**

2. What is printed by the following code? (The variables are integers).

```
#define TRUE   1
#define FALSE  0
...
a = 1;
b = 6;
c = 5;
status = ((c / b) || (a * b)) && ((c - b / a) * c);
printf("%d", status);
```

A. 0
B. 1
C. 5
D. TRUE
E. FALSE
ANSWER: **B**
Widely missed. A surprising number of students chose D or E.

3. Look at the following code, and determine what value is printed.

```
int trace1=0, trace2=0;

for (j=0; j < 4; j = j+1) {
      for (k=0; k < 2; k = k+1) {
            trace1++;
      }
      trace2++;
}
printf ("%d", trace1+trace2);
...
```

A. 4
B. 6
C. 8
D. 12
E. 15

ANSWER: **D**
Missed by quite a few people. A conservative way to solve it is to make four columns, one for each variable, and carefully trace each step. Start out with:

j	k	trace1	trace2
?	?	0	0
0	?	0	0
0	0	0	0
0	0	1	0

etc.
Sometimes you will discover a feature or general pattern which lets you skip ahead to the answer.

4. What is the ouput of the following code? Assume all variables are integers.

```
if ((a > 5) && (a <= 5))
   printf("Sonics\n");
if ( !( (b == 2) || (b != 2) ) )
   printf("Mariners\n");
else
   printf("Seahawks\n");
```

A. Mariners
B. Seahawks
C. Mariners
Seahawks
D. Sonics
Mariners
Seahawks
E. unable to determine: depends on the values of a and b

ANSWER: **B**

5. In the ancient days (summer of 1978), worldwide oil shortages led to long lines at gas stations. Certain professions had special privileges and did not have to wait in these lines. In particular, police officers, firefighters and doctors were given these privileges --but professors were not!.

Read the following code, then choose the correct conditiona so that the code will properly assign privileges to the above mentioned professions.

```
#define POLICE    1
#define DOCTOR  2
#define FIREFIGHTER 3
#define PROFESSOR 4
...
char privileged;
int profession;
  ...

if (       ???       ) /*choose the conditional needed here*/
{
   privileged = 'Y';
}
else
{
   privileged = 'N';
}
...
```

A. profession == POLICE || FIREFIGHTER || DOCTOR

B. profession == POLICE && FIREFIGHTER && DOCTOR

C. profession = POLICE || profession = FIREFIGHTER || profession = DOCTOR && profession != PROFESSOR

D. profession == POLICE || profession == FIREFIGHTER || profession == DOCTOR

E. profession == POLICE && profession == FIREFIGHTER && profession == DOCTOR && profession != PROFESSOR

ANSWER: **D**

6. Study this program, and determine: What would the printf in main produce?

```
void f1 (int i, int j)
{
  i = i + 1;
  j = j - 1;
}

int f2 (int i, int j)
{
  i = i + 1;
  j = j - 1;
  return (i + 2);
}

int main (void)
{
  int i, j;
  i = 4;
  j = 5;
  f1(j, i);
  j = f2(i, j);
  i = f2(j, i);
  f1(i, j);
  printf("i is %d, j is %d\n", i, j);
  return 0;
}
```

A. i is 4, j is 10
B. i is 7, j is 5
C. i is 10, j is 7
D. i is 5, j is 7
E. i is 4, j is 5

ANSWER: **C**

7. What would the following program fragment output?

```
...
int main (void)
{
  int i, j = 1;
  for (i = 3; i > 1; i = i - 1) {
      j = j + 5 * i;
  }

  printf("%d ", j);
}
```

A. 16 26 31
B. 16 26
C. 26
D. 31
E. The loop will not stop running.

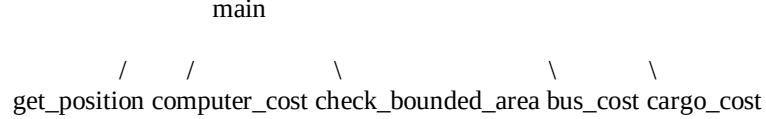
ANSWER: **C**

8. Which call graph is best to describe the following algorithm that helps schedule transportation pick-ups:

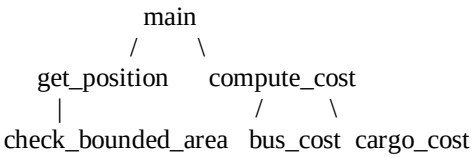
The main program asks the user for the position they want to be picked up at, and then computes the cost. When asking for the position, the program will check to see if the position is in the bounded area. As part of computing the cost, the program asks users for type of transaction (bus, cargo) and uses an appropriate function to compute the cost.

[printf/scanf are omitted from the call graphs for this problem.]

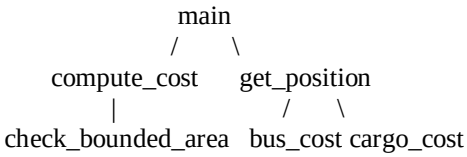
A.



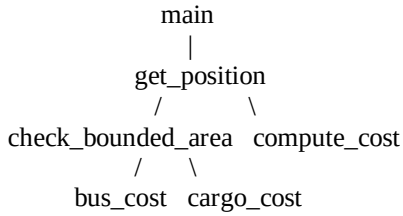
B.



C.



D.



E. None of the above choices even remotely describes the given algorithm.

ANSWER: **B**

9. Here's a truth table that is not completely filled in. A and B stand for any conditional expressions. Fill in the ((!A) || B) column, then pick the matching answer. (Hint: it might help you to add an intermediate column.)

A	B	((!A) || B)
T	T	
T	F	
F	T	
F	F	

A. T
F
T
T

B. F
T
F
F

C. F
T
F
T

D. T
T
F
T

E. F
F
F
F

ANSWER: **A**

10. At /* Point A*/ in the program, what is the value of the variable arrow?

```
...
void converge (int * target) {
    if (*target > 0)
        *target  = *target -1;
    else
        *target = *target +1;
}
...
int arrow=0;
converge (&arrow);
converge (&arrow);
converge (&arrow);
/* Point A */
```

A. -2
B. -1
C. 0
D. 1
E. 2

ANSWER: **D**

11. Given the function definition below, what does the following statement print out?

```
 switchAndPrint(2);

-----------------------------
void switchAndPrint(int x)
{
 switch(x)
 {
 case 1:
   printf("1");
   break;
 case 2:
   printf("2");
 default:
   printf("3");
   break;
 }
}
```

A. 1
B. 2
C. 3
D. 23
E. 123

ANSWER: **D**

12. Study the following function testForDivisors, which is given without comments, but which does in fact have something to do with the divisors of an integer.

What value is returned by this function when it is called with an argument value of 14?

```
int testForDivisors(int x)
{
 int nD = 0;
 int testNumber;

 for(testNumber = 2; testNumber < x; testNumber++) {
    if(x % testNumber == 0)
       nD++;
 }
 return nD;
}
```

A. 0
B. 2
C. 10
D. 11
E. 14

ANSWER: **B**

13. Which C condition captures the meaning of the English-language phrase "P or Q but not both"?

A. (P || !Q) && (!P || Q)
B. (P || !P) && (Q || !Q)
C. (P || Q) || (!P || !Q)
D. (P || Q) && !(P || Q)
E. (P && !Q) || (!P && Q)
ANSWER: **E**

Based on the Question of the Day 4/01/01.
A good way to verify the answer is with a truth table. A = "P or Q but not both" means this:

P	Q	A
T	T	F
T	F	T
F	T	T
F	F	F

14. Examine this code fragment, and determine the value of rain[0] after the last statement shown.

```
int rain[4]; /*declares an array of 4 integers.
   The elements are rain[0], rain[1], rain[2], and rain[3] */
int i;

for (i=0; i < 3; i=i+1) {
    rain[i] = 1;
}
rain[0] = rain[1] + rain[2];
/* What is rain[0] now?? */
```

A. 0
B. 1
C. 2
D. 3
E. 4
ANSWER: **C**

15. In a game program, "bullets" are displayed as moving circles. A function computes new coordinates for the center of a bullet, given its current position and "offset" (distance being moved); the function also tells whether or not the bullet is fully visible on the screen after being moved. The function does no I/O. Chose the function prototype which would be most appropriate for this task.

A. void moveBullet (double radius, int oldX, int oldY, int XOffset, int YOffset, int newY, int newY);

B. int moveBullet (double radius, int oldX, int oldY, int XOffset, int YOffset, int * newX, int * newY);

C. void moveBullet (double radius, int oldX, int oldY, int XOffset, int YOffset, int * newX, int * newY, int isOnScreen);

D. int moveBullet (double radius, int oldX, int oldY, int XOffset, int YOffset, int * newX, int * newY, int isOnScreen);

E. (int newX, int newY, int isOnScreen) moveBullet (double radius, int oldX, int oldY, int XOffset, int YOffset);

ANSWER: **B**

The function needs the old coordinates (a part of ints) and the offsets (another pair of ints). It computes three things: the new coordinates and a Boolean (int) to "tell" about the visibility: 3 ints total. A function can return only one value. One way to give back the three ints would be to return one of them (the Boolean) and pass the others by reference, that is, as int * parameters. (Another reasonable design would be to pass all three as int * params, but that is not one of the choices.)

Some students chose D instead of B. This will unintentionally a bit tricky. B is better, because D has a useless parameter, isOnScreen.

16. [worth 5 M.C. questions] A cryptography algorithm requires a pair of prime integers. To be a "valid pair", two numbers must be prime, must be different from one another, and there must be no primes between the two.

Your task: write a function which, given any two integers, determines whether they constitute a valid pair.

You may not call any functions at all (no, not printf and not scanf!), except the following one:

```
int isPrime (int num);
    /*returns 1 if num is prime, 0 otherwise*/
```

You are not asked to implement isPrime.

Part of your task is to determine appropriate parameters and return type (if any) of the function. Include comments to describe what these are (but you can assume the reader of your code is familiar with the definition of "valid pair"). Use your best style.

Some examples of valid or invalid pairs:
15 17 is not valid, because 15 is not prime.
13 19 is not valid, because there is a prime (17) between 13 and 19.
17 19 is valid.
31 29 is valid
31 31 is not valid (the numbers must be different)

Start your code here and continue on the next page if necessary.

/********************** sample solution ******************/**

```
int isValidPair(int num1, int num2) {
 /*If num1 and num2 constitute a "valid pair",
 the function returns 1.
 Otherwise, the function returns 0.
 */

 int currentNum; /* will be used in the loop*/
 int temp; /*used for swapping*/

 if (num1 == num2)
   return 0;
 if (!(isPrime(num1)))
   return 0;
 if (!(isPrime(num2)))
   return 0;

 if (num1 > num2) {
          /*the input numbers are out of order -- swap them*/
          temp = num1;
          num1 = num2;
          num2 = temp;
   }

 for (currentNum = num1+1;  /*don't check num1 !!*/
              currentNum < num2; /*dont' check num2, either!*/
                    currentNum++) {
```

```
    if (isPrime(currentNum))
        return 0;
    }

 /* If we get here, we have survived all the tests */
 return 1;
}
```

Notes: The first task is to design the function prototype: chose an appropriate name for the function, and give its parameters and return type. The function clear needs two parameters to work on: in the phrase "given any two integers" the word "given" tells you that the integers are parameters. A comment should mention that the two parameters are the numbers being tested to see if they are a valid pair.
The return type is a little more subtle. The function is supposed to "determine" something -- but it can't announce the result of this determination via a printf. So it must give the answer back, either as a return value, or as an output (pointer) parameter. The appropriate type is Boolean ("whether or not" implies "true or false"), but since C doesn't have a Boolean type, you have to use integer. The meaning of the return (or output param) needs to be captured in a comment. [You could instead return a char, with a comment as to what values are possible (probably 'y' or 'n'); experienced C programmers would be VERY unlikely to design it that way.]

The body of the function cannot assume that num1 is less than num2, because the spec says "ANY two integers." So some check is needed to see whether the numbers are in order, and logic to swap them or to otherwise take their order into account. The sample solution swaps them. [If you made an assuption that the first num was less than the second, it would be a design error, but the penalty would be reduced if you documented the precondition with a comment.]

The function should only return true if all of several conditions are met. Namely, num1 and num2 must be different, num1 must be prime; num2 must be prime, and all the values in-between must be non-prime. If a single condition is violated, the function must return false. The easiest way to program this is just to return 0 immediately when a violating condition is found.

What about checking for conditions such as num1 or num2 are not 0, or negative? This is completely unnecessary. The function isPrime knows how to determine primality, so making such a check in your function is not only redundant, but shows a misunderstanding of how isPrime should be used.

A loop is needed to check all the numbers between num1 and num2. The loop should not check num1 or num2 -- only the numbers between them.

The function you write needs absolutely no knowledge of what is or is not a prime number -- all of that is contained in the isPrime function.

Common errors:
-- misinterpreting the task. Some people thought it was "write a program..." instead of "write a function..." There was no need to write a main, for example.

A number of people IMPLEMENTED (defined the code for) isPrime. Doing so earned no points, and took time away from the rest of the task.

A number of people have told us the printf/scanf instructions were confusing. Fortunately, most people did figure them out. If you think you needed printf or scanf in this particular function, it indicates some basic misunderstanding about what parameters or return values are.