**Question 5**

* **What is/are the simple case(s) of recursive function ten?                 int ten(int n){ int ans;**
  + - 1. **if (n < 0) ans = 0;**
      2. **else if (n % 5 == 0) ans = n;                      else if (n % 2 == 0) ans = 1 + ten(n - 1);**
      3. **else ans = 1 + ten(n - 2);                       return (ans);**
      4. **}**
      5. Answer

|  |  |  |
| --- | --- | --- |
|  |  | * when n is negative |
|  |  | * when n is divisible by 5 |
|  |  | * when n is even |
|  | X | * cases a and b |
|  |  | * cases a, b, and c |

**Question 6**

* **Choose statements to complete function print\_digits so it displays the digits of a positive integer in their normal left-to-right order separated by spaces.                 void                print\_digits(int n)                {                      if (n > 0) {                            \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_                            \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_                      }                }** Answer

|  |  |  |
| --- | --- | --- |
|  |  | * print\_digits(n / 10);           printf("%d ", n % 10); |
|  |  | * print\_digits(n % 10);           printf("%d ", n / 10); |
|  | X | * printf("%d ", n % 10);           print\_digits(n / 10); |
|  |  | * printf("%d ", n / 10);           print\_digits(n % 10); |
|  |  | * none of the above |

**Question 8**

**A recursive function can only return a value of type int or double.** Answer   False

**Question 9**

**The code of a function can contain at most one recursive call.** Answer  False

**Question 10**

* **What does function five do?**
* **int five(const char \*str){ int  ans;**
  + - * 1. **if (str[0] == '\0')                             ans = 0;**
        2. **else if (isdigit(str[0]))                      ans = 1 + five(&str[1]);                      else ans = five(&str[1]);                      return (ans);**
        3. **}**
        4. Answer

|  |  |  |
| --- | --- | --- |
|  |  | * counts the number of digits in str |
|  |  | * returns 1 if first character is a digit, 5 otherwise |
|  |  | * counts the number of null characters in str |
|  | X | * counts the number of characters in str |
|  |  | * none of the above |

##### Chapter 9 Recursion

1. A recursive function must have **only one** simple case, and all other cases must reduce to that simple case in a finite number of steps to avoid infinite recursion. [False]
2. A "stack overflow" error often indicates that a recursive function is not terminating. [True]
3. A simple case is an instance of a problem that has a straightforward solution that requires no redefinition of the problem. [True]
4. A recursive function that returns a value must be designed so that every possible path through the function includes a "return" statement. [True]
5. A typical recursive solution will break a problem of size n into n/2 problems of size n-1. [False]
6. For some problems one can specify more natural solutions using recursion than using iteration. [True]
7. When designing a recursive function, one must trust that the function will work before even finishing it. [True]
8. The code of a function can contain at most one recursive call. [False]
9. The terminating condition is \_\_\_\_\_\_.

Int fox(int m, int n)

{

int ans;

if (m < 10)

if (n < 10)

ans = m + n;

else

ans = fox(m, n - 2) + n;

else

ans = fox(m - 1, n) + m;

return (ans);

}

a. m and n equal 10

\*b. m and n are less than 10

c. m is not less than 10

d. m is less than 10

e. n is less than 10

1. What does function six do if it is called with an integer greater than 1?

int

six(int n)

{

int ans;

if (n <= 1)

ans = 1;

else if (n % 2 == 0)

ans = n \* six(n - 2);

else

ans = six(n - 1);

return (ans);

}

a. computes the product of n and n - 2

b. returns n - 1

\*c. computes the product of the even integers between 2 and n inclusive

d. computes the product of the odd integers between 1 and n inclusive

e. none of the above