**Question 19**

* **What value is returned by the following call to strlen?                 strlen("robot")** Answer

|  |  |  |
| --- | --- | --- |
|  |  | * 't' |
|  |  | * 4 |
|  |  | * 5 |
|  |  | * 6 |
|  |  | * none of the above |

##### Chapter 8 Strings

1. The string "Hello" could be stored in the variable str declared as follows

char str[5]; [False]

1. The assignment operator = can be used for string assignment only when initializing a string variable in its declaration. [True]
2. If "cat" is the value of the string variable str, then str[3] is the null character. [True]
3. If name is a string variable whose value is "Elizabeth", the function call

strcpy(target, &name[5]);

assigns the string "beth" to target. [True]

1. The length of a string is determined by the size of the variable in which it is stored and does not depend on the value stored in the variable. [False]
2. The following function call stores in target the null-terminated string "Eliza".

strncpy(target, "Elizabeth", 5); [False]

1. The following code fragment stores "compute" in variable str, and then changes the 'u' to an 'e' so str's value is "compete".

strcpy(str, "compute");

str[4] = 'e'; [True]

1. The condition

str1 < str2

is true if the string value of str1 alphabetically precedes the string value of str2.

[False]

1. The string library function strcmp compares the lengths of two strings. [False]
2. What is the value of variable s after execution of the program fragment below?

char h[6] = "wild";

char p[6] = "crazy";

char s[10];

strcpy(s, h);

strcat(s, p);

a. "wild crazy"

b. "wild craz"

\*c. "wildcrazy"

d. The value of s is undefined.

e. none of the above.

1. What is the value of the expression that follows?

strcmp("5", "49");

a. -1

b. 0

\*c. 1

d. <

e. >

1. What does the following C function do?

int

fun(const char \*string)

{

char blank = ' ';

int k;

int looking = 1;

k = strlen(string);

while (k >= 0 && looking) {

if (string[k] == blank)

--k;

else

looking = 0;

}

if (looking)

return (-1);

else

return (k);

}

a. It finds the subscript of the first nonblank character in string.

\*b. It finds the subscript of the last nonblank character in string.

c. It counts the nonblank characters in string.

d. It finds the subscript of the first blank in string.

e. It finds the subscript of the last blank in string.

1. Consider the following code fragment.

char str[10];

scanf("%s", str);

What will happen if scanf encounters the string "vivaciously" when scanning a value for str?

a. Since there is not enough room in str for the whole string, only "vivacious" will be stored in str.

\*b. Function scanf will store the entire string "vivaciously", even though there is insufficient space in str. The string will overflow str.

c. The program will abort with an error message.

d. Only "vivacious" will be scanned and stored, leaving "ly" on the input line.

e. None of the above.

1. What will be displayed by the statements below?

char s1[8] = "petunia", s2[9] = "marigold";

char tmp1[10], tmp2[20];

strcpy(tmp2, s1);

strcat(tmp2, s2);

strncpy(tmp1, &tmp2[5], 6);

tmp1[6] = '\0';

printf("b%s\n", tmp1);

a. iamari

\*b. biamari

c. oldpet

d. boldpet

e. none of the above

1. In the fragment below, what is the minimum size of result required for successful and valid concatenation of "double " and "trouble"?

strcpy(result, "double ");

strcat(result, "trouble");

a. 10

b. 13

c. 14

\*d. 15

e. none of the above

1. What is displayed by the code fragment below if the memory for NEXT immediately follows the memory for word?

char word[12], NEXT[4] = "Joe";

word[0] = 'c';

word[1] = 'a';

word[2] = 't';

printf("%s\n", word);

a. a line with just the word "cat"

b. a line with the word "cat" followed by nine blanks

\*c. the word "cat" followed by whatever garbage is in word[3] through word[11] followed by Joe (unless there is a null character somewhere in word[3] through word[11])

d. an error message

e. the word "cat", the word "Joe", and the word "cat" again

1. What is accomplished by the call to sprint in the code fragment below? (I got this one wrong) for the ans there is no placeholder, notice than only num and num+10 have placeholder.

char ans[20];

int num= 40;

sprintf(ans, “%d to %d”, num, num+10);

1. Nothing, the fuction name is misspelled.
2. It returns as its value the string “40 to 50”.
3. It displays first the value of ans and then the string “40 to 50” (without the quote marks).
4. It aborts because the value of ans is garbage.
5. None of the above.
6. In the code fragment below, the value assigned to the type double variable x is -2.5.

sscanf(“5 -0.5 end”, “%d%lf%s”, &n, &d, str);

x = n \* d;

[True]

1. What is accomplished by this statement, assuming that ch1 is type char, str1 references a 10-element char array, and n is of type int?

sscanf(“a number 11”, “%c%s%d”, &ch1, str1, &n);

a. Nothing, the fuction name is misspelled.

b. The letter ‘a’ is stored in ch1 and the number 11 is stored in n, but the string “number” is lost because there is no ampersand on the reference to str1.

c. The statement returns the value 2, because it successfully stores values in ch1 and str1, but there is an input conversion error when it attempts to get a value for n.

\*d. The statement stores the letter ‘a’ in ch1, the string “number” in the array referenced by str1, and the integer 11 in n.

e. None of the above.