

Your Name: _____
ID#: _____TA's Name: _____
Section #: _____**KEY: Take Home: Quiz 1 (15 pts) - Review of C Language Concepts**

Using Canvas <https://canvas.wsu.edu/>, please submit your solution to the correct quiz folder. Your solution should be a .pdf file with the name <your last name>_quiz1.pdf and uploaded. To upload your solution, please navigate to your correct Canvas **lab** course space. Select the “Assignments” link in the main left menu bar. Navigate to the correct quiz submission folder. Click the “Start Assignment” button. Click the “Upload File” button. Choose the appropriate .pdf file with your solution. Finally, click the “Submit Assignment” button.

NOTE: The use of gen AI is forbidden for quizzes.

1. (5 pts) Write/Implement a function definition for a function `myStrncpy()` with the following header:

```
char *myStrncpy (char *destination, const char *source, int n)
```

This function copies no more than `n` characters from the string pointed to by `source` to the buffer pointed to by `destination`. If the length of the C string in `source` is less than `n`, then the `destination` is padded with null characters until `n` characters have been copied to it. The function returns `destination`.

```
/* This function copies no more than n characters from the string pointed to by
source to the buffer pointed to by destination. If the length of the C string in
source is less than n, then the destination is padded with null characters until n
characters have been copied to it. The function returns destination. This
implementation uses array notation. */
```

```
char *myStrncpy(char *destination, const char *source, int n)
{
    int index = 0;

    for (index = 0; (source[index] != '\0') && (index < n); ++index)
    {
        // copy one character at a time
        destination[index] = source[index];
    }

    // if fewer than n characters were copied, then
    // assign the null character to pad end of destination
    for (; index < n; ++index)
    {
        destination[index] = '\0';
    }

    return destination;
}
```

CptS 122 – Data Structures



Date Assigned: Fri, Aug 28, 2026 Due Date: Tues, Sep 8, 2026

Your Name: _____
ID#: _____

TA's Name: _____
Section #: _____

2. (5 pts) Write a function `strRevRec ()` which accepts a *pointer* to a string and a *length*, and recursively *reverses* the *characters* in the string. The string reversal has to occur without the involvement of another string or character buffer. This function should return a *pointer* to the reversed string.

```
char* strRevRec(char* str, int length)
{
    char temp = '\0';

    if (length > 1) // there are at least two characters left to swap
    {
        // let's swap the character at the current str pointer with the
        // character at the end of the str based on the current length value
        temp = *str;
        // subtract one to ensure we don't swap the null
        // character or in later recursive steps,
        // character that was already swapped
        *str = *(str + length - 1);
        *(str + length - 1) = temp;

        strRevRec(str + 1, // move to the next character in sequence
                  length - 2); // every time we go to the next character to the
                                // right, the end of the string also must decrease
                                // by 1, hence, length changes by 2
    }

    return str;
}
```


 Your Name: _____
 ID#: _____

 TA's Name: _____
 Section #: _____

3. (5 pts) Write a function called `bubble_sort()` that accepts an *array of strings* (i.e. 2-D array of characters) and the *number* of strings in the array as arguments, and returns nothing. The function sorts the strings in dictionary order according to the following algorithm:

1. set the marker *U* for the unsorted section at the end of the *list* (*U* is an integer index value)
2. while the unsorted section has more than one element do steps 3 through 7
3. set the current element marker *C* at the second element of the *list* (*C* is an integer index value)
4. while *C* has not passed *U* do steps 5 and 6
5. if the item at position *C* is less than the item to its left then exchange these two items (recall based on string comparisons)
6. move *C* to the right one position
7. move *U* left one position
8. stop

```
void bubbleSort(char arrStrs[][100], int size)
{
    char temp[100] = "";
    int U = size - 1; // 1. set U for the unsorted section at the end of the list
    int C = 0; // this is the current element marker

    while (U > 0) // 2. while the unsorted section has more than one
        // element do steps 3 through 7
    {
        C = 1; // 3. set the current element marker C at the second element of the
        // list (C is an integer index value)
        while (C <= U) // 4. while C has not passed U do steps 5 and 6
        {
            // 5. if the item at position C is less than the item to its left
            if (strcmp(arrStrs[C], arrStrs[C - 1]) < 0)
            {
                // then exchange these two items (recall based
                // on string comparisons)
                strcpy(temp, arrStrs[C]);
                strcpy(arrStrs[C], arrStrs[C - 1]);
                strcpy(arrStrs[C - 1], temp);
            }
            ++C; // 6. move C to the right one position
        }
        --U; // 7. move U left one position
    } // 8. stop
}
```