

Exercise 0 (6 points – 1 point per question – No program required)

1. C
2. A
3. E
4. C
5. B
6. B

References and Explanations:

In addition to the course book references cited below, these topics are also covered in the live lectures (in-class students) and the recorded lectures (online students).

1. Notes 1.7, 6.14; The value of any legal expression having the form `++x` is the same as the value of the expression `x + 1`. If `x` has an object pointer type, the sum is not necessarily the result of simply adding the two operands arithmetically. Instead, it is obtained by adding the pointer's value to the value of the product of the integral expression's value and the number of bytes in the type pointed to by the pointer. In this example, where `sizeof(int)` has arbitrarily been defined to be 2, the sum is equal to:
$$20 + (1 * \text{sizeof}(\text{int})) == 20 + (1 * 2) == 20 * 2 == 22$$
2. Notes 6.14, 6.16; Every expression except the expression in answer A is in one of the four equivalent forms `p[i]`, `i[p]`, `*(p + i)`, and `*(i + p)`, where `p` represents an expression having an object pointer type and `i` represents an expression having an integral type. `p` is represented by `abc` and the value of the integral expression is `a`. (Remember that addition is commutative.)
3. Note 5.11; `ip` is an uninitialized automatic variable and as such contains a garbage value. Using a garbage value results in a garbage program, which if you are extremely lucky will result in a program crash but if you are extremely unlucky will appear to work correctly.
4. Notes 6.1, 6.5, 6.16; By the Right-Left Rule the identifier `p` is of type "array of 9 `ints`". Any expression that has an array type and is not used as the sole operand of the `&` or `sizeof` operators is automatically converted to an expression that has type "pointer to type". Therefore, the first argument is a pointer to an `int`. If an expression that has an array type is used as the sole operand of the `&` or `sizeof` operators its data type remains unchanged. Thus, the second argument is a pointer to an array of 9 `ints`, which is the same thing as a "pointer to an array of 9 `ints`".
5. Note 7.3; The input line is: one two three
 - a. The `"\n"` portion of the `scanf` control string reads and throws away characters from the remaining input string up to but not including the first character that is not a whitespace. That character is the `"o"` in `"one"`.
 - b. The `"%31[a-t]"` portion of the `scanf` control string reads characters from the remaining input string up to but not including the first character that is not in the range `"a-t"` and is not a space, up to a maximum of 31 characters. All characters read are stored in array `a` and terminated with the null character, `'\0'`. Thus, the string `"one t"` is stored in array `a`.
 - c. The `"\n"` portion of the `scanf` control string reads and throws away characters from the remaining input string up to but not including the first character that is not a whitespace. That character is `"w"`.
 - d. The `"%[^\t]"` portion of the `scanf` control string reads characters from the remaining input string up to but not including the first character that is a `"t"`. All characters read are stored in array `b` and terminated with the null character, `'\0'`. Thus, the string `"wo "` is stored in array `b`.

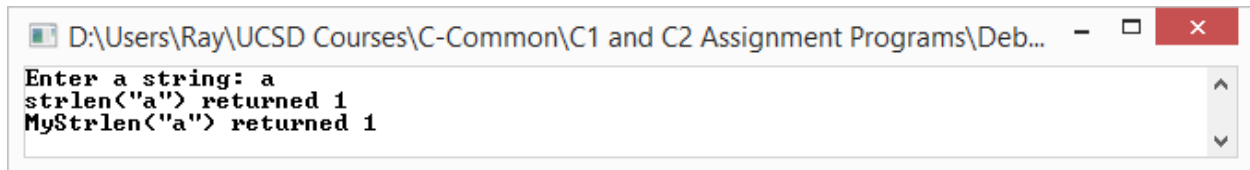
- 1 e. The " " (blank space) portion of the *scanf* control string reads and throws away
2 characters from the remaining input string up to but not including the first character
3 that is not a whitespace. That character is the "t" in "three".
- 4 f. The "%s" portion of the *scanf* control string reads and throws away characters from the
5 remaining input string up to but not including the first character that is not a
6 whitespace. That character is the "t" in "three". It then reads characters from the
7 remaining input string up to but not including the first whitespace. All characters read
8 are stored in array *c* and terminated with the null character, '\0'. Thus, the string
9 "three" is stored in array *c*.
- 10 g. The *scanf* function returns a count of the number of its arguments for which input
11 values were successfully matched assigned, which in this case is 3 (*a*, *b*, and *c*). Thus, 3
12 is assigned into variable *x*.
- 13 h. Finally, the *printf* prints the value of variable *x*, followed by a blank space, followed by
14 the strings stored in arrays *a*, *b*, and *c*, respectively, with no blank space between
15 those strings. The complete output is: 3 one two three
16
- 17 6. Notes 6.14, 6.16, 7.1; The expression *buf* += " of" + " this is: " contains two string literals. A string
18 literal is merely a character array containing a sequence of constant characters ending with the
19 null character that is automatically placed there by the compiler. As with any array, a string
20 literal will decay to a pointer to its first element if not used as either the sole operand of the
21 address operator or the sole operand of the **sizeof** operator. Since the two previous string literals
22 are not being used in either of these two special cases, they each decay to type "pointer to
23 **const char**". Since the previous expression is attempting to add these two pointers together but
24 the addition of two pointers is not one of the three legal arithmetic operations that may be
25 performed on pointers, the compiler will generate an error and will not compile the code.

Exercise 1 (4 points – C Program)

```
1  ***** FILE C1A6E1_MyStrlen.c *****
2
3  /*
4   * ...the usual title block Student/Course/Assignment/Compiler information goes here...
5   *
6   * This file contains function MyStrlen, which duplicates the syntax and
7   * functionality of the standard library strlen function.
8   */
9
10
11 #include <stddef.h>
12 /*
13  * Duplicate the syntax and functionality of the standard library strlen
14  * function. Specifically, a count of the number of characters in the
15  * string represented by pointer <s1> (not including the null terminator
16  * character) is returned.
17  */
18 size_t MyStrlen(const char *s1)
19 {
20     const char *charPtr = s1;
21
22     /* Inspect the characters in <s1> until '\0' is reached. */
23     for (; *s1; ++s1)
24         ;
25     /* Return the number of characters inspected, not including the '\0'. */
26     return (size_t)(s1 - charPtr);
27 }
28
29
30 ***** FILE C1A6E1_main.c *****
31
32 /*
33  * ...the usual title block Student/Course/Assignment/Compiler information goes here...
34  *
35  * This file contains function main, which displays the results of calling the
36  * strlen function and custom MyStrlen function on a user-prompted string.
37  */
38
39 #include <stdio.h>
40 #include <stdlib.h>
41 #include <string.h>
42
43 #define MAX_SIZE 256 /* input buffer size */
44
45 size_t MyStrlen(const char *s1);
46
47 /*
48  * Test the strlen and MyStrlen functions.
49  */
50 int main(void)
51 {
52     char s1[MAX_SIZE];
53     int length;
54
55     printf("Enter a string: ");
56     fgets(s1, MAX_SIZE, stdin);
57     length = (int)strlen(s1);
58     if (length > 0 && s1[length - 1] == '\n')
```

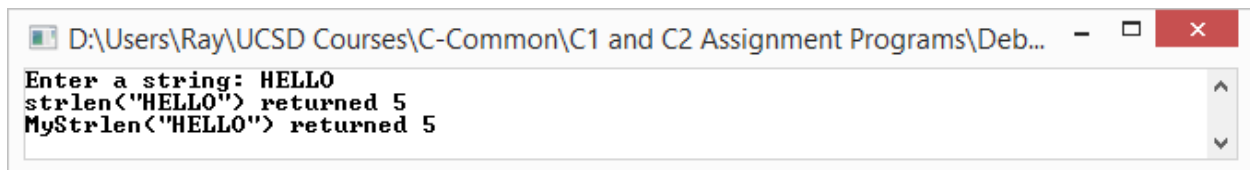
```
1     s1[--length] = '\0';           /* eliminate trailing newline */
2
3     /* Display messages describing the relationship between the strings. */
4     printf("strlen(\"%s\") returned %d\n", s1, length);
5     printf("MyStrlen(\"%s\") returned %d\n", s1, (int)MyStrlen(s1));
6
7     return EXIT_SUCCESS;
8 }
```

C1A6E1 Screen Shots



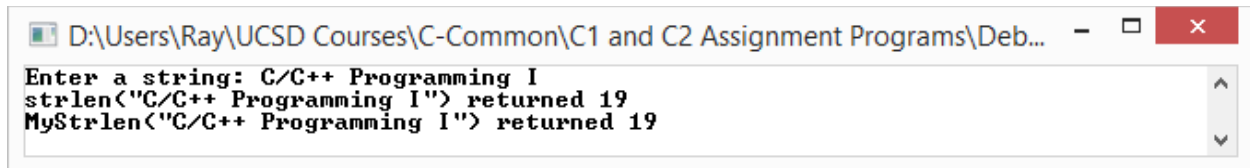
D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb... - □ ×

```
Enter a string: a
strlen("a") returned 1
MyStrlen("a") returned 1
```



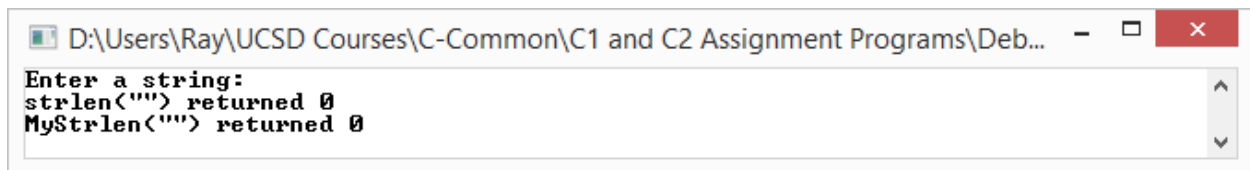
D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb... - □ ×

```
Enter a string: HELLO
strlen("HELLO") returned 5
MyStrlen("HELLO") returned 5
```



D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb... - □ ×

```
Enter a string: C/C++ Programming I
strlen("C/C++ Programming I") returned 19
MyStrlen("C/C++ Programming I") returned 19
```



D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb... - □ ×

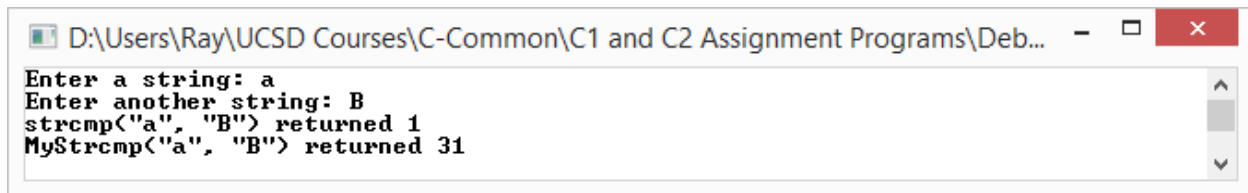
```
Enter a string:
strlen("") returned 0
MyStrlen("") returned 0
```

Exercise 2 (4 points – C Program)

```
1  ***** FILE C1A6E2_MyStrcmp.c *****
2
3  /*
4   * ...the usual title block Student/Course/Assignment/Compiler information goes here...
5   *
6   * This file contains function MyStrcmp, which duplicates the syntax and
7   * functionality of the standard library strcmp function.
8   */
9
10
11 /*
12  * Duplicate the syntax and functionality of the standard library strcmp
13  * function. Specifically, a value is returned as follows:
14  *   <0 if the string represented by pointer <s1> is lexicographically
15  *     less than the string represented by pointer <s2>;
16  *   ==0 if the string represented by pointer <s1> is lexicographically
17  *     equal to the string represented by pointer <s2>;
18  *   >0 if the string represented by pointer <s1> is lexicographically
19  *     greater than the string represented by pointer <s2>;
20  */
21 int MyStrcmp(const char *s1, const char *s2)
22 {
23     /* Compare the strings in <s1> and <s2>. */
24     for (; *s1 == *s2 && *s1; ++s1, ++s2)
25         ;
26     /* Return a value indicating the relationship between the strings. */
27     return (int)*s1 - (int)*s2;
28 }
29
30
31 ***** FILE C1A6E2_main.c *****
32
33 /*
34  * ...the usual title block Student/Course/Assignment/Compiler information goes here...
35  *
36  * This file contains function main, which displays the results of calling the
37  * strcmp function and custom MyStrcmp function on user-prompted strings.
38  */
39
40 #include <stdio.h>
41 #include <stdlib.h>
42 #include <string.h>
43
44 #define MAX_SIZE 256 /* input buffer size */
45
46 int MyStrcmp(const char *s1, const char *s2);
47
48 /*
49  * Test the strcmp and MyStrcmp functions.
50  */
51 int main(void)
52 {
53     char s1[MAX_SIZE];
54     char s2[MAX_SIZE];
55     int length;
56
57     printf("Enter a string: ");
58     fgets(s1, MAX_SIZE, stdin);
```

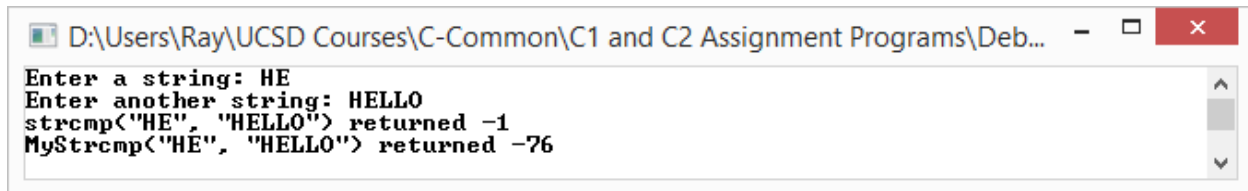
```
1  length = (int)strlen(s1) - 1;
2  if (length >= 0 && s1[length] == '\n')
3      s1[length] = '\0';          /* eliminate trailing newline */
4
5  printf("Enter another string: ");
6  fgets(s2, MAX_SIZE, stdin);
7  length = (int)strlen(s2) - 1;
8  if (length >= 0 && s2[length] == '\n')
9      s2[length] = '\0';          /* eliminate trailing newline */
10
11 /* Display messages describing the relationship between the strings. */
12 printf("strcmp(\"%s\", \"%s\") returned %d\n", s1, s2, strcmp(s1, s2));
13 printf("MyStrcmp(\"%s\", \"%s\") returned %d\n", s1, s2, MyStrcmp(s1, s2));
14
15 return EXIT_SUCCESS;
16 }
```

C1A6E2 Screen Shots



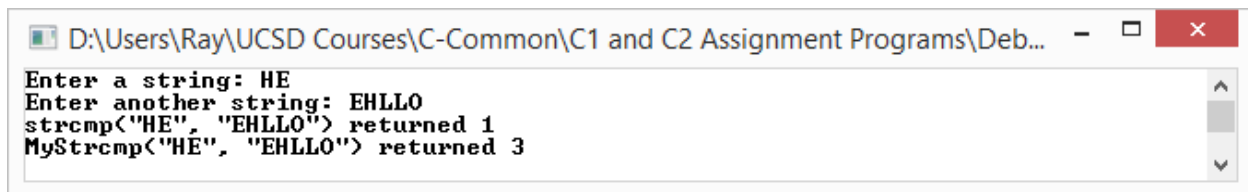
D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb... - □ ×

```
Enter a string: a
Enter another string: B
strcmp("a", "B") returned 1
MyStrcmp("a", "B") returned 31
```



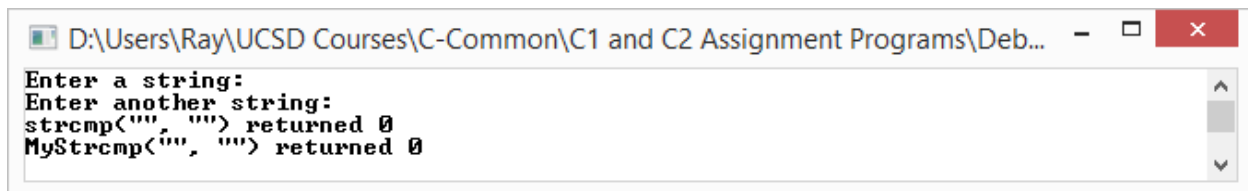
D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb... - □ ×

```
Enter a string: HE
Enter another string: HELLO
strcmp("HE", "HELLO") returned -1
MyStrcmp("HE", "HELLO") returned -76
```



D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb... - □ ×

```
Enter a string: HE
Enter another string: EHLLO
strcmp("HE", "EHLLO") returned 1
MyStrcmp("HE", "EHLLO") returned 3
```



D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb... - □ ×

```
Enter a string:
Enter another string:
strcmp("", "") returned 0
MyStrcmp("", "") returned 0
```

Exercise 3 (6 points – C Program)

***** FILE C1A6E3_GetSubstring.c *****

```
/*
 * ...the usual title block Student/Course/Assignment/Compiler information goes here...
 *
 * This file contains function GetSubstring, which extracts a substring
 * of specified length from a specified position in another string.
 */

/*
 * Copy a substring from the string in <source> into the the array in
 * <result>, terminating it with a null character. <start> specifies
 * the 0-based index in <source> where the substring to be copied starts
 * and <count> represents the number of characters to copy. If <start>
 * represents an index beyond the end of the string in <source> only a
 * null character is placed in the array in <result>. If the value of
 * <count> is such that the substring's length would extend beyond the
 * end of the string in <source>, copying ends at the end of the string
 * in <source>. It is assumed that the array in <result> is large enough
 * to hold the copied substring. The original value of parameter <result>
 * is returned.
 */
```

```
char *GetSubstring(const char source[], int start, int count, char result[])
{
    char *savedTarget = result;    /* save pointer for return */

    while (*source && start--)      /* find start if it's in source */
        ++source;

    while (*source && count--)      /* copy substring */
        *result++ = *source++;

    *result = '\0';                /* null terminate substring */
    return savedTarget;            /* pointer to substring */
}
```

***** FILE C1A6E3_main.c *****

```
/*
 * ...the usual title block Student/Course/Assignment/Compiler information goes here...
 *
 * This file contains function main, which prompts the user for a string and
 * information about extracting a substring from it. Function GetSubstring is
 * called to do the extraction and the substring is displayed by main.
 */
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

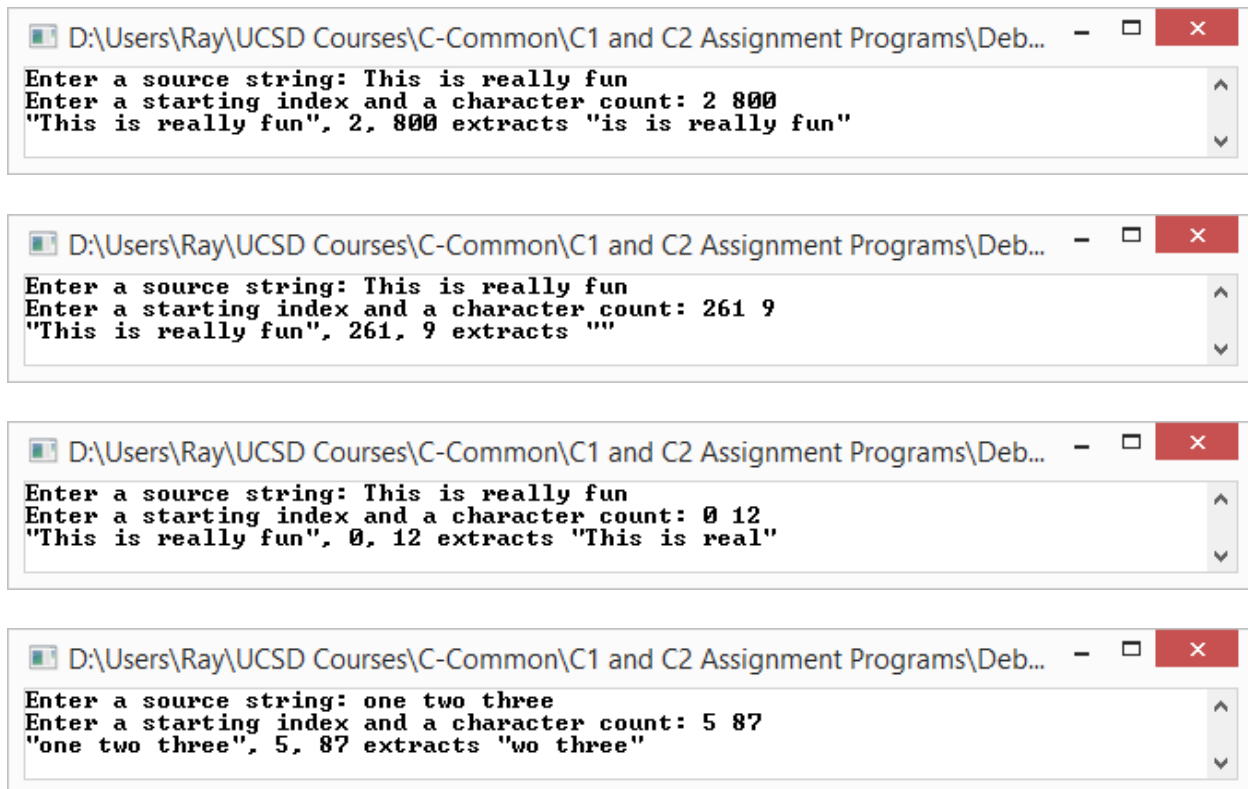
#define MAX_SIZE 256    /* size of input buffers */

char *GetSubstring(const char source[], int start, int count, char result[]);

/*
 * The user is prompted for a string, an index within that string, and a
 * character count. These are passed to function GetSubstring which
 * extracts the specified substring. The substring is displayed by main.
 */
```

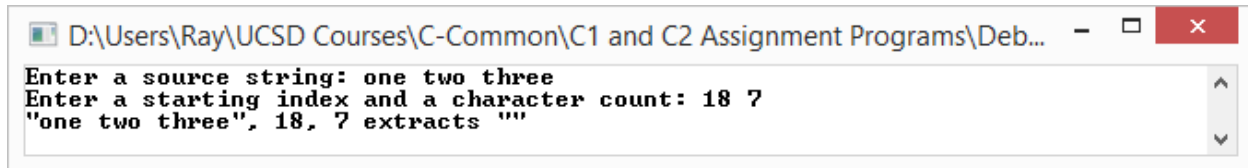
```
1 int main(void)
2 {
3     char source[MAX_SIZE];
4     char result[MAX_SIZE];
5     int start, count, length;
6
7     printf("Enter a source string: ");
8     fgets(source, MAX_SIZE, stdin);
9     length = (int)strlen(source) - 1;
10    if (length >= 0 && source[length] == '\n')
11        source[length] = '\0';          /* eliminate trailing newline */
12
13    printf("Enter a starting index and a character count: ");
14    scanf("%d %d", &start, &count);
15    printf("\n\"%s\", %d, %d extracts \"%s\"\n", source, start, count,
16          GetSubstring(source, start, count, result));
17
18    return EXIT_SUCCESS;
19 }
```

C1A6E3 Screen Shots

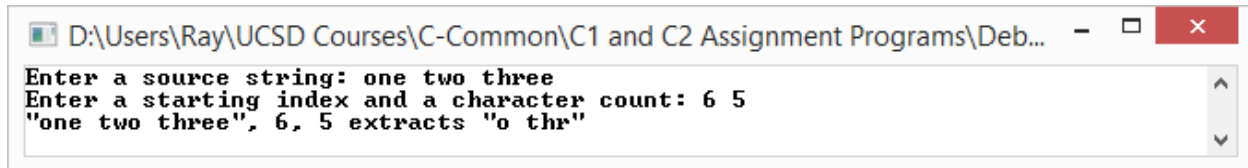


C1A6E3 Screen Shots continue on the next page...

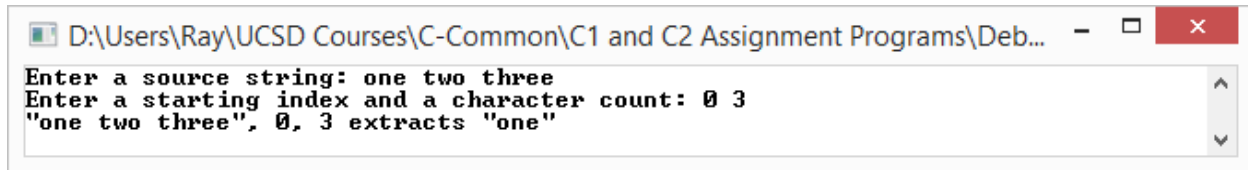
C1A6E3 Screen Shots, continued



A screenshot of a Windows application window titled "D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...". The window contains a text area with the following text: "Enter a source string: one two three", "Enter a starting index and a character count: 18 7", and "one two three", 18, 7 extracts ""



A screenshot of a Windows application window titled "D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...". The window contains a text area with the following text: "Enter a source string: one two three", "Enter a starting index and a character count: 6 5", and "one two three", 6, 5 extracts "o thr"



A screenshot of a Windows application window titled "D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...". The window contains a text area with the following text: "Enter a source string: one two three", "Enter a starting index and a character count: 0 3", and "one two three", 0, 3 extracts "one"