# Assignment 5
## C/C++ Programming I

1
2
3

4 **C1A5 General Information**

5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

24 # --- No General Information for This Assignment---

25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

47 **Get a Consolidated Assignment 5 Report (optional)**

48 If you would like to receive a consolidated report containing the results of the most recent version of
49 each exercise submitted for this assignment, send an empty email to the assignment checker with the
50 subject line **C1A5_ID**, where **ID** is your 9-character UCSD student ID.  Inspect the report carefully since it is
51 what I will be grading.  You may resubmit exercises and report requests as many times as you wish
52 before the assignment deadline.

## C1A5E0 *(6 points total - 1 point per question – No program required)*

Assume language standards compliance and any necessary support code unless stated otherwise. Testing erroneous or implementation dependent code by running it can be misleading. These <u>are not</u> trick questions and each has only one correct answer. Major applicable course book notes are listed.

1. The declaration *int (\*(\*x)())();* in English is: (Note 6.1)
   A. "x is a function returning a pointer to a function returning a pointer to an int"
   B. "x is a pointer to a function returning a pointer to an int"
   C. "x is a pointer to a function returning a pointer to a function returning an int"
   D. "x is a function returning a pointer to a function returning an int"
   E. This is not a valid declaration!

2. Which is the most likely output from:
   > **const int** save[] = {1, 2, 3, 4};
   > **for** (**int** index = 0; index < 6; ++index)
   >     cout << save[index] << ' ';

   (Note 6.2)
   A. *1 2 3 4 5 6*
   B. *1 2 3 4* then two garbage values and/or a system error occurs
   C. *1 2 3 4 5* then a garbage value and/or a system error occurs
   D. *1 2 3 4 0 0*
   E. *1 2 3 4*

3. What is the most important problem with these two successive lines of code?
   > **double** \*p;
   > \*p = 47E3;

   (Notes 6.6 & 6.7)
   A. *p* is initialized to a null pointer by default.
   B. *p* is uninitialized.
   C. Pointers to **double**s are not reliable in standard C/C++.
   D. 47E3 is not a legal address.
   E. There is no significant problem!

4. For:
   > **int** x = 5, y = 10;
   > fcnA(&x, &y);

   (Note 6.9)
   A. **int** types are being passed.
   B. C++ reference types are being passed.
   C. The types of *x* and *y* are implementation dependent.
   D. *fcnA* can't change the value of *x* or *y*.
   E. Pointer types are being passed

5. If *fcnA* only does *return(\*iP1 + \*iP2)* what, if anything, is seriously wrong with:
   > **int** fcnA(**int** \*iP1, **const int** \*iP2);
   > **const int** x = 5, y = 10;
   > fcnA(&x, &y);

   (Note 6.10)
   A. *const int \** is legal in C++ but not C.
   B. *int const \*iP2* is illegal in C and C++
   C. The right argument causes an error.
   D. The left argument causes an error.
   E. There are no reference types in C.

6. What is wrong with:
   > **int** \*fcnA(**int** y)
   > {
   >     **int** x = y;
   >     **return**(&x);
   > }

   (Note 6.12)
   A. An automatic variable is returned.
   B. A reference to an automatic variable is returned.
   C. A pointer to an automatic variable is returned.
   D. **return**(&x) should be **return**(\*x)
   E. Nothing is wrong.

## Submitting your solution

Using the format below place your answers in a plain text file named **C1A5E0_Quiz.txt** and send it to the assignment checker with the subject line **C1A5E0_ID**, where **ID** is your 9-character UCSD student ID.

> *-- Place an appropriate "Title Block" here --*
> 1. A
> 2. C
> etc.

*See the course document titled "How to Prepare and Submit Assignments" for additional exercise formatting, submission, and assignment checker requirements.*

1  <mark>C1A5E1 *(6 points – C Program)*</mark>
2  Exclude any existing source code files that may already be in your IDE project and add a new one,
3  naming it **C1A5E1_main.c**.  Write a program in that file to implement a survey that prompts respondents
4  to enter a decimal integer value within a specified range to indicate how much they like a product.
5  The survey will end and the results will be displayed when either a specific number of legal responses
6  have been entered or a specific number of <u>consecutive</u> out-of-range responses have been entered.
7
8  Define the following four macros and use them appropriately in your code.  Respectively, use values
9  3, -3, -3, and 3 for your first test, but your program must also work for any other permitted values.  You
10 may define additional macros if you wish but it must not be necessary to explicitly change their values if
11 the values of any of the four required macros are changed.
12     `MAX_RESPONDENTS` – maximum number of respondents  (> 0)
13     `MIN_RESPONSE_VALUE` – lowest response value  (any positive or negative value)
14     `MAX_RESPONSE_VALUE` – highest response value  (>= `MIN_RESPONSE_VALUE`)
15     `OUT_OF_RANGE_LIMIT` – number of <u>consecutive</u> out-of-range responses that end survey  (> 0)
16
17 The code in function **main** must conduct the survey, then display the results when the survey ends as
18 follows:
19     1.  Collect legal user responses in a 1-dimensional automatic array having exactly the number of
20         elements as there are possible legal response values.  For example, if `MIN_RESPONSE_VALUE` is `-2`
21         and `MAX_RESPONSE_VALUE` is `3` there would be `6` elements.  No other arrays are permitted.  Each
22         element represents a unique legal response value and when the survey ends will contain a
23         count of how many of that value have occurred (similar to note 6.3).
24     2.  Use a variable named `consecutiveRangeErrors` to count the number of consecutive
25         out-of-range responses:
26         • If an illegal response occurs increment, then test that variable.  If the maximum count has
27           not yet been reached display an error message and re-prompt the same user; if it has
28           been reached end the survey.
29         • If a legal response occurs set the variable back to 0 and increment the appropriate array
30           element.  End the survey if all respondents have entered legal responses.
31     3.  Once the survey has ended display a table <u>in descending rating order</u> like the one below.  It
32         indicates the total number of responses for each possible rating.  DO NOT put any blank lines or
33         other dividers between table entries.  The least significant digits of all values in their respective
34         columns must be aligned for readability:
35

```
        Rating         Responses
        -------        ---------
           10              25
            9              50
            8             100
           ...             ...
        -1239               0
```

44
45     • If you believe you need more than one array, or an array having `MAX_RESPONDENTS` elements,
46       you have not understood the requirements and should reread them as well as note 6.3.
47     • <u>Do not</u> test if the values of the four required macros are actually valid – just assume they are.
48     • <u>Do not</u> use a loop or call a function to initialize your array to 0s.
49     • <u>Do not</u> produce the absolute value of anything.  It's totally unnecessary.
50
51 Manually re-run your program several times with at least the sets of macro values shown on the next
52 page, using a sufficient assortment of user input values to verify functionality.  To test with different
53 macro values you will need to recompile after each change.

| Test Value Sets | | | | |
|---|---|---|---|---|
| Test Set | MAX_ RESPONDENTS | MIN_RESPONSE_ VALUE | MAX_RESPONSE_ VALUE | OUT_OF_RANGE_ LIMIT |
| A | 3 | -3 | -3 | 3 |
| B | 3 | 0 | 0 | 4 |
| C | 20 | 3 | 15 | 2 |
| D | 5 | -100 | -86 | 3 |
| E | 17 | -27 | 9 | 1 |

## Submitting your solution

Send your source code file to the assignment checker with the subject line **C1A5E1_ID**, where **ID** is your 9-character UCSD student ID.

*See the course document titled "How to Prepare and Submit Assignments" for additional exercise formatting, submission, and assignment checker requirements.*

**Hint:**

See note 6.3 for a similar exercise.  Whenever appropriate the best way to initialize an automatic array to all 0s is by using the standard syntax:

```
int ratingCounters[RESPONSE_VALUES] = {0};
```

Be sure the array contains the correct number of elements.  For example, if **MIN_RESPONSE_VALUE** were **-3** and **MAX_RESPONSE_VALUE** were **7**, the number of elements required would be **11**.  Also, be sure your algorithm can support negative response values and response ranges that do not start or end with **0**. This can easily be accomplished by always subtracting **MIN_RESPONSE_VALUE** from the user input response value when using it as an index into the array.  Finally, be sure you reset your "bad response" count to its initial value each time a correct response is received.

## C1A5E2 *(4 points – C++ Program)*

Exclude any existing source code files that may already be in your IDE project and add three new ones, naming them **C1A5E2_ComputeMinimum.cpp**, **C1A5E2_ComputeMaximum.cpp**, and **C1A5E2_main.cpp**.  <u>Do not</u> use **#include** to include any of these three files in each other or in any other file.  However, you may use it to include any appropriate header file(s) you need.

File **C1A5E2_ComputeMinimum.cpp** must contain a function named **ComputeMinimum** and **C1A5E2_ComputeMaximum.cpp** must contain a function named **ComputeMaximum**.  Each function must:
1. have exactly two formal parameters, each of type "reference to **const double**";
2. return type "reference to **double**" (not "reference to **const double**");
3. contain only one statement;
4. <u>not</u> use variables other than its formal parameters;
5. <u>not</u> use anything that requires **#define** or **#include**;
6. <u>not</u> use literal values;
7. <u>not</u> do assignment, addition, subtraction, multiplication, or division;
8. <u>not</u> use **if**, **switch**, or looping statements;
9. <u>not</u> call functions or macros;
10. <u>not</u> display anything.

**ComputeMinimum** must compare the values referenced by its parameters and return a reference to the smallest of those values whereas **ComputeMaximum** must compare the values referenced by its parameters and return a reference to the greatest of those values.

File **C1A5E2_main.cpp** must contain function **main**, which must:
1. use no more than two variables;
2. prompt the user to enter two space-separated decimal values on the same line;
3. pass references to the user-entered values to both **ComputeMinimum** and **ComputeMaximum** as arguments;
4. display the results of both function calls using the following 2-line format, where the question marks represent the values whose references are passed to and returned from the functions:
```
ComputeMinimum(?, ?) returned ?
ComputeMaximum(?, ?) returned ?
```
For example, if the user enters **-5.8 5.8** the result should be:
```
ComputeMinimum(-5.8, 5.8) returned -5.8
ComputeMaximum(-5.8, 5.8) returned 5.8
```

- Do <u>not</u> attempt to detect cases where the user input values are equal.  Instead, simply treat them exactly like any other values.
- Scientific and standard notation are both okay and may be mixed.
- Zeros that don't affect a fractional part's value may be omitted.
- If a fractional part is empty the decimal point may be omitted.

Manually re-run your program several times, testing with at least the following 5 sets of user input values, where each set represents the argument values in left-to-right order:
```
6.9 6.4      6.4 6.9      -5.8 5.8      -0.0 0.0      8.4e3 6.2e-1
```

### Submitting your solution

Send your three source code files to the assignment checker with the subject line **C1A5E2_ID**, where *ID* is your 9-character UCSD student ID.

*See the course document titled "How to Prepare and Submit Assignments" for additional exercise formatting, submission, and assignment checker requirements.*

1
2    **Hints:**
3    Use the conditional operator to both compare the values referenced to by each parameter and
4    produce the reference that references the greatest value.  Return the entire conditional expression
5    (Note 3.16), type casting it as (***double &***) to override the "**const**-ness".  (Same principle as in note 6.12).

1    ## C1A5E3 *(4 points – C++ Program)*

2    Exclude any source code existing files that may already be in your IDE project and add three new ones,
3    naming them **C1A5E3_ComputeMinimum.cpp**, **C1A5E3_ComputeMaximum.cpp**, and
4    **C1A5E3_main.cpp**. <u>Do not</u> use `#include` to include any of these three files in each other or in any other
5    file.  However, you may use it to include any appropriate header file(s) you need.

6
7    File **C1A5E3_ComputeMinimum.cpp** must contain a function named `ComputeMinimum` and
8    **C1A5E3_ComputeMaximum.cpp** must contain a function named `ComputeMaximum`.  Each function
9    must:
10       1.  have exactly two formal parameters, each of type "pointer to **const double**";
11       2.  return type "pointer to **double**" (not "pointer to **const double**");
12       3.  contain only one statement;
13       4.  <u>not</u> use variables other than its formal parameters;
14       5.  <u>not</u> use anything that requires `#define` or `#include`;
15       6.  <u>not</u> use literal values;
16       7.  <u>not</u> do assignment, addition, subtraction, multiplication, or division;
17       8.  <u>not</u> use `if`, `switch`, or looping statements;
18       9.  <u>not</u> call functions or macros;
19       10. <u>not</u> display anything.
20   `ComputeMinimum` must compare the values pointed to by its parameters and return a pointer to the
21   smallest of those values whereas `ComputeMaximum` must compare the values pointed to by its
22   parameters and return a pointer to the greatest of those values.

23
24   File **C1A5E3_main.cpp** must contain function `main`, which must:
25       1.  use no more than two variables;
26       2.  prompt the user to enter two space-separated decimal values on the same line;
27       3.  pass pointers to the user-entered values to both `ComputeMinimum` and `ComputeMaximum` as
28           arguments;
29       4.  display the results of both function calls using the following 2-line format, where the question
30           marks represent the values whose pointers are passed to and returned from the functions and
31           the ampersands are displayed literally to remind the user that pointers are being passed and
32           returned, not the values themselves:
33           **`ComputeMinimum(&?, &?) returned &?`**
34           **`ComputeMaximum(&?, &?) returned &?`**
35           For example, if the user enters **-5.8 5.8** the result should be:
36           **`ComputeMinimum(&-5.8, &5.8) returned &-5.8`**
37           **`ComputeMaximum(&-5.8, &5.8) returned &5.8`**

38
39   •   Do <u>not</u> attempt to detect cases where the user input values are equal.  Instead, simply treat them
40       exactly like any other values.
41   •   Scientific and standard notation are both okay and may be mixed.
42   •   Zeros that don't affect a fractional part's value may be omitted.
43   •   If a fractional part is empty the decimal point may be omitted.

44
45   Manually re-run your program several times, testing with at least the following 5 sets of user input values,
46   where each set represents the argument values in left-to-right order:
47       **6.9 6.4        6.4 6.9        -5.8 5.8        -0.0 0.0        8.4e3 6.2e-1**
48
49

50   ## Submitting your solution

51   Send your three source code files to the Assignment Checker with the subject line **C1A5E3_ID**, where **ID**
52   is your 9-character UCSD student ID.

---

1   *See the course document titled "Preparing and Submitting Your Assignments" for additional exercise*
2   *formatting, submission, and Assignment Checker requirements.*
3

---

4   **Hints:**
5   Use the conditional operator to both compare the values pointed to by each parameter and produce
6   the pointer that points to the greatest value.  Return the entire conditional expression (Note 3.16), type
7   casting it as *(double \*)* to override the "**const**-ness".  (Note 6.12).