Lab 5: Recursion

Thomas Bischoff

Original

Date: 3 / 26 / 2018

Problem 1: character\_replacer.cpp

The function of this program will be to take a string entered by the user and then allow them to swap out any character they want with a different character.

Pseudocode 1: character\_replacer.cpp

int main() –

1. Initialize a variable to contain the string entered by the user of the program.
2. Set the variable to hold the value returned by the function user\_provided\_string().
3. Run the function character\_changer(string str), where str represents the string contained by the variable initialized in step one of this function.

string user\_provided\_string() –

1. Initialize a variable to hold the user’s string they will input.
2. Display a message asking the user to enter a string into the function.
3. Take the string entered by the user and set it to be held by the variable initialize in step one of this function.
4. Return the variable holding the user’s string from the function.

void character\_changer(string str) –

1. Initialize a variable to hold the character that the user wants to change.
2. Display a message asking the user to enter in the character that they wish to replace into the function.
3. Take the user’s input and set it to be held by the variable initialized in step one of this function.
4. Check if the user’s character is in the string by running the function search(string str, char character), where str is the string entered by the user and character is the character held by the variable initialized in step one of the function.

Pseudocode 1: character\_replacer.cpp

void character\_changer(string str) –

1. If search(string str, char character) returns ‘true’.
2. Initialize a variable to hold the character that is going to replace the character which the user specified that they want replaced.
3. Display a message asking the user to input the character the character that they want to add.
4. Take the user’s entered character and set it to be held by the variable initialized in step i.
5. Create a variable which is equal to the length of the user’s string.
6. Go through each character in the string.

* Check if the current character is the type that the user wants to replace.

1. If the two characters are the same.

* Replace the current character with the one that the user wishes to add.

1. If they are different, just continue.
2. Display the newly updated string.
3. Check if the user wants to change another character by running the function run\_again().

* If run\_again() returns ‘true’.

1. Re-run the function character\_changer(string str) where str is the updated string.

* If run\_again() returns ‘false’, terminate the program.

1. If search(string str, char character) returns ‘false’.
2. Re-run the function character\_changer(string str), where str is the same string.

bool search(string str, char character) –

1. Initialize a variable that is equal to the length of the user’s string.
2. Go through each character in the string.
3. Check if the current character is the same as the character that the user is looking for.
4. If they are the same, return ‘true’.

Pseudocode 1: character\_replacer.cpp

bool search(string str, char character) –

1. Display a message indicating that character that they are looking for is not in the string and that the program will be re-run.
2. Return ‘false’.

bool run\_again() –

1. Initialize a statement to be ‘true’.
2. Initialize a statement to hold the user’s decision if they want to continue.
3. Create a ‘while’ loop which will continue as long as the statement initialized in step one of this function is ‘true’.
4. Initialize a variable to hold the character indicating the user’s response to if they want to continue or not.
5. Display a message asking if the user wants to continue using the program by entering in either a ‘Y’ for yes or an ‘N’ for no.
6. Take the user’s response and set it to be held by the variable initialized in step A.
7. Convert the character held by the variable to be uppercase.
8. Check to see if the variable is the same as ‘Y’.
9. If they are the same.

* Change the statement initialized in step one of this function to be ‘false’.
* Set the statement initialized in step two of this function to be ‘true’.

1. If they are not the same continue.
2. Check to see if the variable is the same as ‘N”.
3. If they are the same.

* Change the statement initialized in step one of this function to be ‘false’.
* Display a goodbye message to the user.
* Set the statement initialized in step two of this function to be ‘false’.

1. If they are not the same, do the following.

Pseudocode 1: character\_replacer.cpp

Code 1: character\_replacer.cpp

bool run\_again() –

* Display a message indicating that the user’s input was not recognized and that it will ask the user the question again.

1. Return the statement initialized in step two of the function containing the user’s response.
2. //######################################################################
3. // Program Header: character\_replacer.cpp
4. // The function of this program is to take a string entered from the
5. // user and then be able to swap out letters with different ones that
6. // the user selects.
7. // Author: Thomas Bischoff
8. // Date Created: 3/9/2018
9. //######################################################################
11. #include <iostream>
12. #include <string>
13. #include <cctype>
14. **using** **namespace** std;
16. // Function Declarations
17. string user\_provided\_string();
18. **void** character\_changer(string str);
19. **bool** search(string str, **char** character);
20. **bool** run\_again();
22. **int** main()
23. {
24. // Initialize a Value to Contain the User Entered String
25. string users\_string;
26. // Set users\_string to be Equal to the String Returned by the Function
27. users\_string = user\_provided\_string();
28. // Run character\_changer
29. character\_changer(users\_string);
30. }
32. // Function Definition
33. string user\_provided\_string()
34. {
35. // Initialize a Value to Take the User's Input
36. std::string users\_input;
37. // Ask the User to Input a String into the Program
38. std::cout << "Please Enter a String into the Program: ";
39. // Take the Input and Set it Equal to users\_input
40. std::getline(std::cin, users\_input);
41. // Return users\_input
42. **return** users\_input;
43. }
45. // Function Definition
46. **void** character\_changer(string str)
47. {
48. // Initialize a Value for the Character That is Going to be Replaced
49. **char** replaced\_character;
50. // Display a Message Asking for the User to Input the Character they Wish to Replace
51. cout << "What Character Would You Like to Replace? ";
52. // Take the Input and Set it Equal to replaced\_character
53. cin >> replaced\_character;
54. // Check if the Character is in the String Using search
55. **if** (search(str, replaced\_character))
56. {
57. // Initialize a Value for the Character that is Going to be Added in
58. **char** added\_character;
59. // Display a Message Asking the User to What Character they Would Like to Add
60. cout << "What Character Would You Like to Take its Place? ";
61. // Take the Input and Set it Equal to added\_character
62. cin >> added\_character;
63. // Create a Value for the Length of the String
64. **int** length\_of\_string = str.length();
65. // Go Through Each Character in the String
66. **for** (**int** i = 0; i < length\_of\_string; i++)
67. {
68. // Check if the Character is the One the User Seeks to Replace
69. **if** (str[i] == replaced\_character)
70. {
71. // Subsitute that Character with the New Character
72. str[i] = added\_character;
73. }
74. }
75. // Display the New String
76. cout << "Updated String: " << str << endl;
77. // Check if the User Wants to use the Program Again Using the Function run\_again
78. **if** (run\_again())
79. {
80. // Re-Run the Program
81. character\_changer(str);
82. }
83. }
84. **else**
85. {
86. // Re-Run the Program
87. character\_changer(str);
88. }
89. }
91. // Function Definition
92. **bool** search(string str, **char** character)
93. {
94. // Initialize a Value for the Length of the String
95. **int** length\_of\_string = str.length();
96. // Go Through Each Character in the String
97. **for** (**int** i = 0; i < length\_of\_string; i++)
98. {
99. // If Character is in the String Return True
100. **if** (str[i] == character)
101. {
102. **return** **true**;
103. }
104. }
105. // Display that the Character Cannot be Found and that the Program Will be Re-Ran
106. cout << "Sorry, but " << character << " Cannot be Found in the String the Program Will Restart." << endl;
107. // Return False
108. **return** **false**;
109. }
111. // Function Definition
112. **bool** run\_again()
113. {
114. // Intiliaze a Statement as True
115. **bool** repeat = **true**;
116. // Initialize a Value to Hold the User's Decision
117. **bool** users\_decision;
118. // Generate a While Loop
119. **while** (repeat)
120. {
121. // Initialize a Value to Hold the User's Response
122. **char** users\_choice;
123. // Ask the User if they Would Like to Run the Program Again
124. cout << "Would You Like to Run This Program Again (Y/N): ";
125. // Take the Users Input and Set it Equal to users\_choice
126. cin >> users\_choice;
127. // Convert the users\_choice Character to Uppercase
128. users\_choice = toupper(users\_choice);
129. // Check the User's Answer
130. **if** (users\_choice == 'Y')
131. {
132. // Change repeat to false
133. repeat = **false**;
134. // Set User's Decison to True
135. users\_decision = **true**;
136. }
137. **else** **if** (users\_choice == 'N')
138. {
139. // Change repeat to false
140. repeat = **false**;
141. // Display a Goodbye Message to the User
142. cout << "Thank You, for Using character\_replacer.cpp" << endl;
143. // Set User's Decision to False
144. users\_decision = **false**;
145. }
146. **else**
147. {
148. // Display that User's Input was Not Recognized and to Try Again
149. cout << "Sorry, but Your Input Cannot be Recognized. Please Try Again." << endl;
150. }
151. }
152. // Retrun users\_decision
153. **return** users\_decision;
154. }

Sample Output 1: character\_replacer.cpp

Problem 2: string\_reverse.cpp

Please Enter a String into the Program: hello world What Character Would You Like to Replace? l What Character Would You Like to Take its Place? t Updated String: hetto wortd Would You Like to Run This Program Again (Y/N): y What Character Would You Like to Replace? a Sorry, but a Cannot be Found in the String the Program Will Restart. What Character Would You Like to Replace? o What Character Would You Like to Take its Place? x Updated String: hettx wxrtd Would You Like to Run This Program Again (Y/N): n Thank You, for Using character\_replacer.cpp

The function of this program is the order of any string entered so that the first character becomes the last and so on.

Pseudocode 2: string\_reverse.cpp

int main() –

1. Initialize a variable to hold the string entered by the user.
2. Set the variable initialized in step one to hold the string returned from user\_provided\_string().
3. Initialize a variable and set it equal to the length of string that the user provided.
4. Initialize a variable indicating the starting position of the string and set it equal to ‘0’.
5. Convert the user’s string to become a character array.
6. Run the function reverser(string &str, char\* character\_list, int starting, int size).
7. Display the reversed form of the string.

string user\_provided\_string() –

1. Initialize a variable to hold the string that the user provides.
2. Display a message asking the user to input a string into the program.

Pseudocode 2: string\_reverse.cpp

Code 2: strirng\_reverse.cpp

string user\_provided\_string() –

1. Take the user’s input and set it to be held by the variable initialized in step one of this function.
2. Return the variable holding the user’s string.

void reverser(string &str, char\* character\_list, int starting, int size) –

1. Check if the starting value is less than or equal to the size of the string.
2. If it is.
3. Set the character to switch positions with the character in its mirrored position in the string.
4. Re-run the function reverser(string &str, char\* character\_list, int starting, int size), where the variable starting is increased by one.
5. //######################################################################
6. // Program Header: string\_reverse.cpp
7. // The function of this program is to switch the order of a given string
8. // from forwards to backwards.
9. // Author: Thomas Bischoff
10. // Date Created: 3/6/2018
11. //######################################################################
13. #include <iostream>
14. #include <string>
15. #include <cctype>
16. **using** **namespace** std;
18. // Function Declaration
19. string user\_provided\_string();
20. **void** reverser(string &str, **char**\* character\_list, **int** starting, **int** size);
22. **int** main()
23. {
24. // Intialize a Value Called User's String
25. string users\_string;
26. // Set User's String to be Equal to the String Returned by the Function
27. users\_string = user\_provided\_string();
28. // Intialize a Value Called Length and Set it Equal to the Length of the String
29. **int** length = users\_string.length();
30. // Intialize the Starting Location
31. **int** start = 0;
32. // Convert the String to a Character Array
33. **char** character\_array[length];
34. // Run Reverser
35. reverser(users\_string, character\_array, start, length);
36. // Display the Reversed String
37. cout << "Reversed String: " << character\_array << endl;
38. }
40. // Function Definition
41. string user\_provided\_string()
42. {
43. // Initialize a Value Called User's Input
44. string users\_input;
45. // Ask the User to Input a String
46. cout << "Please Enter in a String: ";
47. // Take the User's Input and Set Equal to User's Input
48. cin >> users\_input;
49. // Return User's Input
50. **return** users\_input;
51. }
53. // Function Definition
54. **void** reverser(string &str, **char**\* character\_list, **int** starting, **int** size)
55. {
56. // Check if starting is Less Than or Equal to size
57. **if** (starting <= size)
58. {
59. character\_list[starting] = str[(size - starting - 1)];
60. reverser(str, character\_list, (starting + 1), size);
61. }
62. }

Sample Output 2: string\_reverse.cpp

Problem 3: upper\_to\_lower.cpp

Please Enter in a String: computer Reversed String: retupmoc

Please Enter in a String: table Reversed String: elbat

Please Enter in a String: paragraph Reversed String: hpargarap

The function of this program is to both convert a string entered by the user to be all uppercase and then all lowercase.

Pseudocode 3: upper\_to\_lower.cpp

int main() –

1. Initialize a variable to hold the string entered by the user.
2. Set the variable initialized in step one of the function to hold the string returned by the function user\_proivided\_string().
3. Initialize a variable to represent the starting location of the string and set it equal to ‘0’.
4. Initialize a variable to be equal to the length of the string minus one.
5. Run the function uppercase(string str, int start, int end), where str is the string that user provided, start is the variable initialized in step three of this function, and end is the variable initialized in step four of this function.
6. Run the function lowercase(string str, int start, int end), where str is the string that user provided, start is the variable initialized in step three of this function, and end is the variable initialized in step four of this function.

string user\_provided\_string() –

1. Initialize a variable to hold the string that the user will provide.
2. Display a message asking for the user to enter a string into the program.
3. Take the user’s string and set it to be held by the variable initialized in step one of the function.
4. Return the variable holding the string that the user provided.

void uppercase(string str, int start, int end) –

1. Check if the value of start is less than or equal to the value of end.
2. If it is.
3. Convert the character located at the current position that start is set as to be uppercase.
4. Re-run the function uppercase(string str, int start, int end), where the value for start is increased by one.
5. If it is not.
6. Display the string in uppercase.

Pseudocode 3: upper\_to\_lower.cpp

Code 3: upper\_to\_lower.cpp

void lowercase(string str, int start, int end) –

1. Check if the value of start is less than or equal to the value of end.
2. If it is.
3. Convert the character located at the point indicated by the value of start to be lowercase.
4. Re-run the function lowercase(string str, int start, int end), where the value of start is increased by one.
5. If it is not.
6. Display the lowercase string.
7. //######################################################################
8. // Program Header: upper\_to\_lower.cpp
9. // The function of this program is to allow the user to convert a string
10. // from being lowercase to uppercase and vis versa.
11. // Author: Thomas Bischoff
12. // Date Created: 3/9/2018
13. //######################################################################
15. #include <iostream>
16. #include <string>
17. #include <cctype>
18. **using** **namespace** std;
20. // Function Declations
21. string user\_provided\_string();
22. **void** uppercase(string str, **int** start, **int** end);
23. **void** lowercase(string str, **int** start, **int** end);
25. **int** main()
26. {
27. // Create a Value to Hold the User's String
28. string users\_string;
29. // Set users\_string to be Equal to the String Returned by the Function
30. users\_string = user\_provided\_string();
31. // Create a Value to be the Starting Location
32. **int** starting\_location = 0;
33. // Create a Value to be the Length of the String
34. **int** length\_of\_string = users\_string.length() - 1;
35. // Run uppercase
36. uppercase(users\_string, starting\_location, length\_of\_string);
37. // Run lowercase
38. lowercase(users\_string, starting\_location, length\_of\_string);
39. }
41. // Function Definition
42. string user\_provided\_string()
43. {
44. // Initialize a Value to Take the User's Input
45. std::string users\_input;
46. // Ask the User to Input a String into the Program
47. std::cout << "Please Enter a String into the Program: ";
48. // Take the Input and Set it Equal to users\_input
49. std::getline(std::cin, users\_input);
50. // Return users\_input
51. **return** users\_input;
52. }
54. // Function Definition
55. **void** uppercase(string str, **int** start, **int** end)
56. {
57. // Check if start is Less Than or Equal to end
58. **if** (start <= end)
59. {
60. // Covert the Character to Uppercase
61. str[start] = toupper(str[start]);
62. // Re-Run the Function
63. uppercase(str, start + 1, end);
64. }
65. **else**
66. {
67. // Display the String All Uppercased
68. cout << "Uppercase: " << str << endl;
69. }
70. }
72. // Function Definition
73. **void** lowercase(string str, **int** start, **int** end)
74. {
75. // Check if start is Less Than or Equal to end
76. **if** (start <= end)
77. {
78. // Covert the Character to Lowercase
79. str[start] = tolower(str[start]);
80. // Re-Run the Function
81. lowercase(str, start + 1, end);
82. }
83. **else**
84. {
85. // Display the String All Lowercase
86. cout << "Lowercase: " << str << endl;
87. }
88. }

Sample Output 3: upper\_to\_lower.cpp

Please Enter a String into the Program: AsSiGnMeNt Uppercase: ASSIGNMENT Lowercase: assignment

Problem 4: recursive\_binary\_search.cpp

The function of this program is to perform a binary search through an array by using a recursive binary search algorithim.

Pseudocode 4: recursive\_binary\_search.cpp

int main() –

1. Initialize a constant integer for the size of the array and set it equal to ‘10’.
2. Allocate dynamic memory for the array.
3. Go through each position in the array.
4. Display a message asking the user to input a value for that position in the array.
5. Take the user’s input and set it to be held in that position of the array.
6. Run the function bubble\_sort(int \*ptr\_array, int length), where the length is equal to the size minus one.
7. Go through and display the sorted array.
8. Initialize a variable to hold the value that the user is searching for.
9. Display a message asking the user to input integer that they wish to find.
10. Take the user’s provided integer and set it to be held by the variable initialized in step six.
11. Run the function binary\_search(int \*ptr\_array, int start, int end, int users\_value), where start is ‘0’, end is the size minus one, and users\_value is the value held by the variable initialized in step six.

void bubble\_sort(int \*ptr\_array, int length) –

1. Initialize a variable and set it to be equal to one.
2. Initialize a variable to represent the last index of the array and set it equal to the value of length.
3. Generate a ‘while’ loop which will repeat as long as the value of the variable initialized in step one is greater than ‘0’.
4. Set the variable initialized in step one to be equal to zero.
5. Go through the array.
6. Check if the value contained at an index in the array is greater than the value contained in the index after it.

Pseudocode 4: recursive\_binary\_search.cpp

void bubble\_sort(int \*ptr\_array, int length) –

* If it is.

1. Increase the variable initialized in step one by one.
2. Initialize a variable to act as a placeholder and set it equal to element located at the lower index.
3. Set the element with lower index to be equal to element with the higher index.
4. Set the element with the higher index to be equal to the value in the placeholder.
5. Decrease the variable initialized in step two by one.

void binary\_search(int \*ptr\_array, int start, int end, int users\_value) –

1. Check if the value held by start is less than or equal to the value held by end.
2. If it is.
3. Initialize a variable to represent the midpoint of the array and set it equal to value represented by start and end divided by two.
4. Check if value contained in the middle of the array is the same as the value contained in users\_value.

* If it is.

1. Display that the integer is located at the current index.

* If not continue to the next statement.

1. Check if value is contained at the middle of the array is less than the value of users\_value.

* If it is.

1. Re-run the function binary\_search(int \*ptr\_array, int start, int end, int users\_value), where end is now equal to the middle plus one.

* If it is not do the following.

1. Re-run the function binary\_search(int \*ptr\_array, int start, int end, int users\_value), where end is now equal to the middle minus one.

Pseudocode 4: recursive\_binary\_search.cpp

Code 4: recursive\_binary\_search.cpp

void binary\_search(int \*ptr\_array, int start, int end, int users\_value) –

1. If it is not.
2. Display that the value that the user is looking for is not in the array.
3. //######################################################################
4. // Program Header: recursive\_binary\_search.cpp
5. // The function of this program is to perform binary search recursively.
6. // Author: Thomas Bischoff
7. // Date Created: 3/9/2018
8. //######################################################################
10. #include <iostream>
11. #include <string>
12. #include <cctype>
13. **using** **namespace** std;
15. // Function Declations
16. **void** bubble\_sort(**int** \*ptr\_array, **int** length);
17. **void** binary\_search(**int** \*ptr\_array, **int** start, **int** end, **int** users\_value);
19. **int** main()
20. {
21. // Initialize a Constant Integer for the Size of the Array
22. **const** **int** SIZE = 10;
23. // Allocate Dyanmic Memory for the Array
24. **int** \*array = **new** **int**[SIZE];
25. // Fill Out the Array
26. **for** (**int** i = 0; i < SIZE; i++)
27. {
28. // Display that the User has to Input a Integer into the Array
29. cout << "Enter in a Integer for Index " << i << ": ";
30. // Set the Element in the Array to the User's Input
31. cin >> array[i];
32. }
33. // Run bubble\_sort
34. bubble\_sort(array, (SIZE - 1));
35. // Display the Sorted Array
36. cout << "Sorted Array:" << endl;
37. **for** (**int** j = 0; j < SIZE; j++)
38. {
39. cout << array[j] << " ";
40. }
41. cout << endl;
42. // Initialize a Value for What the User is Searching for
43. **int** search\_value;
44. // Ask the User to Input a Integer that They Want to Find
45. cout << "What Integer are You Looking for? ";
46. // Take the User's Input and Set it Equal to search\_value
47. cin >> search\_value;
48. // Run binary\_search
49. binary\_search(array, 0, (SIZE - 1), search\_value);
50. }
52. // Function Definition
53. **void** bubble\_sort(**int** \*ptr\_array, **int** length)
54. {
55. **int** nr\_swaps\_per\_pass = 1;
56. **int** index\_end = length;
57. **while** (nr\_swaps\_per\_pass > 0)
58. {
59. nr\_swaps\_per\_pass = 0;
60. **for** (**int** i = 0; i < index\_end; i++)
61. {
62. **if** (ptr\_array[i] > ptr\_array[i + 1])
63. {
64. nr\_swaps\_per\_pass++;
65. // Swap ptr\_array[i] with ptr\_array[i + 1]
66. **int** tmp = ptr\_array[i];
67. ptr\_array[i] = ptr\_array[i + 1];
68. ptr\_array[i + 1] = tmp;
69. }
70. }
71. index\_end --;
72. }
73. }
75. // Function Definition
76. **void** binary\_search(**int** \*ptr\_array, **int** start, **int** end, **int** users\_value)
77. {
78. // Check if the Starting Value is Less Than or Eqaul to the Ending Value
79. **if** (start <= end)
80. {
81. // Initialize and Calculate the Mid-Point
82. **int** middle = (start + end) / 2;
83. // Check if the Integer Contained in the Middle of the Array is the One That the User is Looking for
84. **if** (ptr\_array[middle] == users\_value)
85. {
86. // Display that the Value is Located at Middle
87. cout << "The Interger '" << users\_value << "' is Located at the Index of '" << middle << "'." << endl;
88. }
89. // Check if the Integer Contained at the Middle is Less than the Value the User is Looking for
90. **else** **if** (ptr\_array[middle] < users\_value)
91. {
92. // Re-Run binary\_search
93. binary\_search(ptr\_array, (middle + 1), end, users\_value);
94. }
95. // Otherwise
96. **else**
97. {
98. // Re-Run binary\_search
99. binary\_search(ptr\_array, start, (middle - 1), users\_value);
100. }
101. }
102. // Otherwise, Display that the Value is Not in the Array
103. **else**
104. {
105. cout << "Sorry, but the Integer '" << users\_value << "' is Not in the Array." << endl;
106. }
107. }

Sample Output 4: recursive\_binary\_search.cpp

Enter in a Integer for Index 0: 5 Enter in a Integer for Index 1: 1 Enter in a Integer for Index 2: -9 Enter in a Integer for Index 3: 34 Enter in a Integer for Index 4: 2 Enter in a Integer for Index 5: 14 Enter in a Integer for Index 6: 67 Enter in a Integer for Index 7: 3 Enter in a Integer for Index 8: 8 Enter in a Integer for Index 9: 7 Sorted Array: -9 1 2 3 5 7 8 14 34 67 What Integer are You Looking for? 14 The Integer ‘14’ is located at the Index of ‘7’.

Problem Notes 4:

The function bubble\_sort(int \*ptr\_array, int length) is based off of an algorithm provided in a previous lab.

Bonus Problem 2: recursive\_permutation.cpp

The function of this program is to determine all the possible ways the numbers between one and ‘n’ can be arranged using a recursive program.

Bonus Pseudocode 2: recursive\_permutation.cpp

int main() –

1. Initialize a variable to hold the value that the user want to increase to from one.
2. Display a message asking the user to provide a positive integer.
3. Take the integer provided by the user and set it to be held by the variable initialized in step one.
4. Initialize a variable to represent the number of ways that the numbers can be sorted and set it equal to one to start.
5. Run the function calculate\_orders(int orders, int max\_number, int original\_number)

Bonus Pseudocode 2: recursive\_permutation.cpp

Bonus Code 2: recursive\_permutation.cpp

void calculate\_orders(int orders, int max\_number, int original\_number) –

1. Check if the value held by max\_number is greater than one.
2. If it is.
3. Calculate the value of orders by setting orders to be equal to orders time max\_number.
4. Re-run the function calculate\_order(int orders, int max\_number, int original\_number), where max\_number has been decreased by one.
5. If it is not do the following.
6. Display the number of ways that the numbers can be arranged.
7. //######################################################################
8. // Program Header: recursive\_permutation.cpp
9. // The function of this program is to output the number of posible ways
10. // that numbers 1 to n can be ordered.
11. // Author: Thomas Bischoff
12. // Date Created: 3/12/2018
13. //######################################################################
15. #include <iostream>
16. **using** **namespace** std;
18. // Function Declarations
19. **void** calculate\_orders(**int** orders, **int** max\_number, **int** original\_number);
21. **int** main()
22. {
23. // Initialize a Value to Indicate What One Increases to
24. **int** largest\_number;
25. // Ask the User to Input an Integer for largest\_number
26. cout << "Please, Enter a Positive Integer: ";
27. // Take the User's Input and Set it Equal to largest\_number
28. cin >> largest\_number;
29. // Initalize a Value for the Number of Ways the Numbers Can be Ordered
30. **int** number\_of\_orders = 1;
31. // Run calcualte\_orders
32. calculate\_orders(number\_of\_orders, largest\_number, largest\_number);
33. }
35. // Function Definition
36. **void** calculate\_orders(**int** orders, **int** max\_number, **int** original\_number)
37. {
38. // Check if max\_number is Greater Than One
39. **if** (max\_number > 1)
40. {
41. // Calculate the New Value for orders
42. orders = orders \* max\_number;
43. // Re-Run calculate\_orders, With New orders and max\_number
44. calculate\_orders(orders, (max\_number - 1), original\_number);
45. }
46. // Otherwise
47. **else**
48. {
49. // Display the Number of Ways it can be Ordered
50. cout << "The Number of Ways '1' to " << original\_number << " can be Ordered is " << orders << endl;
51. }
52. }

Bonus Sample Output 2:

Please, Enter a Positive Integer: 6 The Number of Ways ‘1’ to 6 can be Ordered is 720.