| **Software Engineering Department - ITU** |
| --- |
| **SE101T: Programming Fundamentals Lab** |

| **Course Instructor: Usama Bin Shakeel** | **Dated: 23/10/2023** |
| --- | --- |
| **Teaching Assistant: Abeera Ashraf** | **Semester: Fall 2023** |
| **Teaching Assistant: Aasma Waheed** | **Batch: BSSE2023** |

# **Lab 8. Arrays in C++**

| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case of ambiguity seek help from the course instructor, lab engineers, assigned teaching assistants

**Lab Tasks**

**Task 1:** Accept the assignment posted in Google Classroom and after accepting clone the repository to your computer for this ensure you have logged into github app with your account.

**Task 2:** Solve the given problems written after task instructions, write code through IDE like CLion

**Task 3:** Ensure your code/solution is in the cloned folder.

**Task 4:** Commit and Push the changes through the Github App

**Task 5:** Run ‘make run’ to run C++ code

Write code in functions named q1, q2, and so on, after completing each part, verify through running code using **“make run”** on cygwin

***Q1. Write a program in a function named isPalindrome() that determines whether the given string is palindrome or not. For example, rar is a palindrome. Take the string input from the user in main () and pass it as an argument to the function.***

| #include <iostream>  #include <string>  #include <cctype>  bool isPalindrome(const std::string &str) {  // Remove spaces and convert to lowercase for case-insensitive comparison  std::string cleanedStr;  for (char ch : str) {  if (!std::isspace(ch)) {  cleanedStr += std::tolower(ch);  }  }  // Check if the cleaned string is a palindrome  int start = 0;  int end = cleanedStr.length() - 1;  while (start < end) {  if (cleanedStr[start] != cleanedStr[end]) {  return false; // Not a palindrome  }  start++;  end--;  }  return true; // Palindrome  }  int main() {  std::string inputString;  // Take input from the user  std::cout << "Enter a string: ";  std::getline(std::cin, inputString);  // Check if the entered string is a palindrome  if (isPalindrome(inputString)) {  std::cout << "The entered string is a palindrome." << std::endl;  } else {  std::cout << "The entered string is not a palindrome." << std::endl;  }  return 0;  } |
| --- |

***Q2. Write a program in a function named Upper\_half() which takes a two dimensional array A, with size N rows and N columns as argument and prints the upper half of the array.***

***e.g.,  
2 3 1 5 0 2 3 1 5 0  
7 1 5 3 1 1 5 3 1  
2 5 7 8 1 Output will be: 1 7 8  
0 1 5 0 1 0 1  
3 4 9 1 5 5***

| #include <iostream>  const int N = 4; // Change this value based on the size of your array  void Upper\_half(int arr[N][N]) {  for (int i = 0; i < N; ++i) {  for (int j = 0; j < N; ++j) {  // Print only the upper half of the array (j >= i)  if (j >= i) {  std::cout << arr[i][j] << " ";  } else {  std::cout << " "; // Print spaces for the lower half  }  }  std::cout << std::endl;  }  }  int main() {  // Example two-dimensional array  int array[N][N] = {{1, 2, 3, 4},  {5, 6, 7, 8},  {9, 10, 11, 12},  {13, 14, 15, 16}};  // Print the upper half of the array  Upper\_half(array);  return 0;  } |
| --- |

***Q3. Write a program in a function named indexShiftting, that asks for an index and a number. Then it includes the number at the indicated index of the array and moves a position forward (from u to u+1) each element after the selected index.***

*Eg. Array is ={1,2,3,4,5,6} with this array index 2 and number 7 is given this should update the array to following values. {1,2,7,3,4,5}*

| #include <iostream>  const int ARRAY\_SIZE = 6; // Change this value based on the size of your array  void indexShifting(int arr[], int size, int index, int number) {  if (index < 0 || index >= size) {  std::cout << "Invalid index." << std::endl;  return;  }  // Shift elements to the right starting from the end of the array  for (int i = size - 1; i > index; --i) {  arr[i] = arr[i - 1];  }  // Include the number at the indicated index  arr[index] = number;  }  int main() {  int array[ARRAY\_SIZE] = {1, 2, 3, 4, 5, 6};  // Display the original array  std::cout << "Original Array: {";  for (int i = 0; i < ARRAY\_SIZE; ++i) {  std::cout << array[i];  if (i < ARRAY\_SIZE - 1) {  std::cout << ", ";  }  }  std::cout << "}" << std::endl;  // Get index and number from the user  int index, number;  std::cout << "Enter the index: ";  std::cin >> index;  std::cout << "Enter the number: ";  std::cin >> number;  // Perform index shifting  indexShifting(array, ARRAY\_SIZE, index, number);  // Display the updated array  std::cout << "Updated Array: {";  for (int i = 0; i < ARRAY\_SIZE; ++i) {  std::cout << array[i];  if (i < ARRAY\_SIZE - 1) {  std::cout << ", ";  }  }  std::cout << "}" << std::endl;  return 0;  } |
| --- |

***Q4. Write a function named "number\_of\_matches" that compares the initial parts of two character arrays to see how many pairs of cells match before a difference occurs. For example, if the arrays are***

******

***then the function should return the value 3 because only the first three pairs of cells in the arrays match (cell 0 matches cell 0, cell 1 matches cell 1, and cell 2 matches cell 2).***

| #include <iostream>  #include <cstring>  int number\_of\_matches(const char arr1[], const char arr2[]) {  int matches = 0;  int length1 = std::strlen(arr1);  int length2 = std::strlen(arr2);  int min\_length = (length1 < length2) ? length1 : length2;  for (int i = 0; i < min\_length; ++i) {  if (arr1[i] == arr2[i]) {  matches++;  } else {  break; // Stop counting when a difference is encountered  }  }  return matches;  }  int main() {  const char array1[] = "abcde";  const char array2[] = "abxyz";  int matches = number\_of\_matches(array1, array2);  std::cout << "Number of matching pairs: " << matches << std::endl;  return 0;  } |
| --- |

### Assessment Rubric for Lab

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & Github Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_