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

| **Course Instructor: Usama Bin Shakeel** | **Dated: 30/10/2023** |
| --- | --- |
| **Teaching Assistant: Abeera Ashraf** | **Semester: Fall 2023** |
| **Teaching Assistant: Aasma Waheed** | **Batch: BSSE2023** |

# **Lab 8. Pointers & File handling**

| **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 q1 to create and store information in a text file of a student (roll number, name & phone number).***

| #include <iostream>  #include <fstream>  #include <string>  struct Student {  int rollNumber;  std::string name;  std::string phoneNumber;  };  void q1() {  // Create an object of the Student struct  Student student;  // Get student information from the user  std::cout << "Enter Roll Number: ";  std::cin >> student.rollNumber;  std::cin.ignore(); // Clear the newline character from the buffer  std::cout << "Enter Name: ";  std::getline(std::cin, student.name);  std::cout << "Enter Phone Number: ";  std::getline(std::cin, student.phoneNumber);  // Open a text file for writing  std::ofstream outFile("student\_info.txt", std::ios::app); // ios::app to append to the file if it already exists  // Check if the file is opened successfully  if (!outFile.is\_open()) {  std::cerr << "Error opening the file." << std::endl;  return;  }  // Write student information to the file  outFile << "Roll Number: " << student.rollNumber << std::endl;  outFile << "Name: " << student.name << std::endl;  outFile << "Phone Number: " << student.phoneNumber << std::endl;  outFile << "-------------------------" << std::endl;  // Close the file  outFile.close();  std::cout << "Student information has been successfully stored in the file." << std::endl;  }  int main() {  // Call the function to store student information in the file  q1();  return 0;  } |
| --- |

***Q2. Write a program in a function named q3 to count the number of words and characters in a file.***

| #include <iostream>  #include <fstream>  #include <string>  #include <sstream>  void q3(const std::string& filename) {  // Open the file for reading  std::ifstream inFile(filename);  // Check if the file is opened successfully  if (!inFile.is\_open()) {  std::cerr << "Error opening the file." << std::endl;  return;  }  std::string line;  int wordCount = 0;  int charCount = 0;  // Read the file line by line  while (std::getline(inFile, line)) {  // Count characters  charCount += line.length();  // Count words using a stringstream  std::istringstream iss(line);  std::string word;  while (iss >> word) {  wordCount++;  }  }  // Close the file  inFile.close();  // Display the results  std::cout << "Number of words in the file: " << wordCount << std::endl;  std::cout << "Number of characters in the file: " << charCount << std::endl;  }  int main() {  // Call the function with the filename as an argument  q3("example.txt");  return 0;  } |
| --- |

***Q3. Write a program in a function named q2 to compute the sum of all elements in an array of user size using pointers.***

| #include <iostream>  void q2() {  int size;  // Get the size of the array from the user  std::cout << "Enter the size of the array: ";  std::cin >> size;  // Dynamically allocate memory for the array  int\* array = new int[size];  // Get array elements from the user  std::cout << "Enter the elements of the array:" << std::endl;  for (int i = 0; i < size; ++i) {  std::cout << "Element " << i + 1 << ": ";  std::cin >> array[i];  }  // Compute the sum using pointers  int sum = 0;  int\* ptr = array; // Initialize pointer to the beginning of the array  for (int i = 0; i < size; ++i) {  sum += \*ptr; // Dereference the pointer to get the value  ptr++; // Move the pointer to the next element  }  // Display the sum  std::cout << "Sum of elements in the array: " << sum << std::endl;  // Deallocate the memory  delete[] array;  }  int main() {  // Call the function to compute the sum of elements in the array  q2();  return 0;  } |
| --- |

***Q4. Write a function named q4 to search for a specific element in a user defined 2D array through pointers and DMA (Dynamic Memory Allocation) and return its row and column indices where the element is found.***

| #include <iostream>  void q4(int\*\* array, int rows, int cols, int element, int& foundRow, int& foundCol) {  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  if (array[i][j] == element) {  foundRow = i;  foundCol = j;  return; // Element found, exit the function  }  }  }  // If the element is not found, set indices to -1  foundRow = -1;  foundCol = -1;  }  int main() {  int rows, cols;  // Get the dimensions of the array from the user  std::cout << "Enter the number of rows: ";  std::cin >> rows;  std::cout << "Enter the number of columns: ";  std::cin >> cols;  // Dynamically allocate memory for the 2D array  int\*\* array = new int\*[rows];  for (int i = 0; i < rows; ++i) {  array[i] = new int[cols];  }  // Get array elements from the user  std::cout << "Enter the elements of the array:" << std::endl;  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  std::cout << "Element at position [" << i << "][" << j << "]: ";  std::cin >> array[i][j];  }  }  int element;  // Get the element to search for from the user  std::cout << "Enter the element to search for: ";  std::cin >> element;  int foundRow, foundCol;  // Call the function to search for the element in the array  q4(array, rows, cols, element, foundRow, foundCol);  // Display the result  if (foundRow != -1 && foundCol != -1) {  std::cout << "Element found at position [" << foundRow << "][" << foundCol << "]." << std::endl;  } else {  std::cout << "Element not found in the array." << std::endl;  }  // Deallocate the memory  for (int i = 0; i < rows; ++i) {  delete[] array[i];  }  delete[] array;  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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_