|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE101L: Object Oriented Programming Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 10/03/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 1B. Getting Familiar with Advance Concepts of GitHub**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| Fatima Ehsan | BSCE21016 |  |  |  |

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

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

## **Objective**

The objective of this lab is to practice problems related to pointers and dynamic memory allocation.

## **Equipment and Component**

|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

**Branches** allow you to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository. You always create a branch from an existing branch. Typically, you might create a new branch from the default branch of your repository.

**Pull** requests let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

**Lab Task**

**Task A (For 1st member of Group)**

**Part 1)** A double pointer is used for declaring two dimensional arrays dynamically. For example,

We want to implement a triangular 2D array in which each row has one column greater than the previous one. i.e., the first row has one column, the second one has two columns, and the third one has three columns and so on. You have to take the total number of rows from the user. Following is an example of a triangular 2D array with four rows.

Box and whisker chart

Description automatically generated

Write following functions with given prototype:

**a. void AddColumns (int \* &, int size)**

This function takes a single pointer by reference and dynamically allocates memory to it. You will call this function in main to allocate number of columns to each row turn by turn.

**int main( ){**

**int \*\*p;**

**p=newint \*[rows];**

**for(int i=0; i<rows; i++)**

**//call function...**

**}**

**b. void RowWiseInput(int \*, int size)**

This function simply takes a pointer as an argument and takes input in it from the user. The second argument is the size of 1D array pointed by pointer.

**c. void RowWisePrint(int \*, int size)**

This function takes a pointer as argument and prints its contents. The second argument is the size of 1D array pointed by pointer.

**d. int main()**

In main function you have to do the following tasks:

* Ask the user to enter the number of rows.
* Declare a 2D array.
* Allocate memory for its columns in this array using **AddColumns** Function defined above.
* Take input in the 2D array using **RowWiseInput** function.
* Print the 2D array using **RowWisePrint** function.
* You may use loops in main.

|  |
| --- |
|  |

**Task A (For 1st member of Group)**

**Part 2)** Do above task opposite with Dynamic Memory Allocation

Diagram

Description automatically generated with medium confidence

|  |
| --- |
| void addColumns(int \*&ptr, int size) {  ptr = new int [size]; }  void rowWiseInput(int \*p, int size) {  cout<<"enter input"<<endl;  for(int i=0; i<size; i++)  {  cin>>p[i];  } }  void rowWisePrint(int \*p, int size) {  for(int i=0; i<size; i++)  {  cout<<p[i]<<" ";  }  cout<<endl; }  **MAIN.CPP**  cout<<"\n-------------TASK 2---------------\n"; int size; cout<<"enter number of rows: "<<endl; cin>>size;  //allocation of 2D array int \*\*p = new int \*[size]; for(int i=0; i<size; i++) {  addColumns(\*p,(size-i)); } for(int i=0; i<size; i++) {  rowWiseInput(\*p,(size-i)); } cout<<"\n-------DISPLAYING ARRAY-----------\n"; for(int i=0; i<size; i++) {  rowWisePrint(\*p, (size-i)); }  //deleting the 2D array for(int i=0; i<size; i++) {  delete [] p[i]; } delete [] p;  **OUTPUT:**  -------------TASK 2---------------  enter number of rows:  4  enter input  1  1  1  1  enter input  1  1  1  enter input  1  1  enter input  1  -------DISPLAYING ARRAY-----------  1 1 1 1  1 1 1  1 1  1  1 |

**Task B**

Commit a code by creating more than one branches

**Task C**

Using a pull request to merge changes to master/parent/base branch

#### **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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_