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

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 08/03/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 1A. Solving Problems by Utilization of Pointers and Dynamic Memory Allocation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| NIMRA MAQBOOL | BSCE21012 |  |  |  |

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**

The variable that is used to hold the memory address of another variable is called a pointer variable or simply a pointer. The data type of the variable (whose address a pointer is to hold) and the pointer variable must be the same. A pointer variable is declared by placing a asterisk (\*) after data type or before variable name in data type statement. E.g. if pointer variable “p” is to hold memory address of an integer variable it is declared as:

**int \*p;**

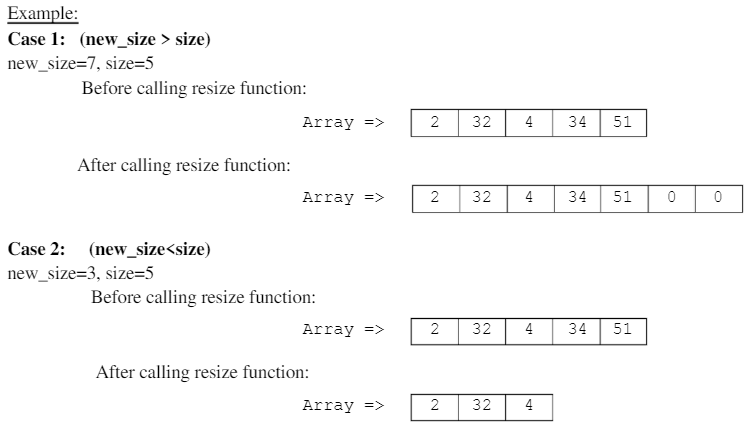
or to hold address of a float type variable we can declare as:

**float \*rep;**

A dynamic array is quite similar to a regular array, but its size is modifiable during program runtime. Dynamic Array elements occupy a contiguous block of memory. Once an array has been created, its size cannot be changed. However, a dynamic array is different. A dynamic array can expand its size even after it has been filled. During the creation of an array, it is allocated a predetermined amount of memory. This is not the case with a dynamic array as it grows its memory size by a certain factor when there is a need.

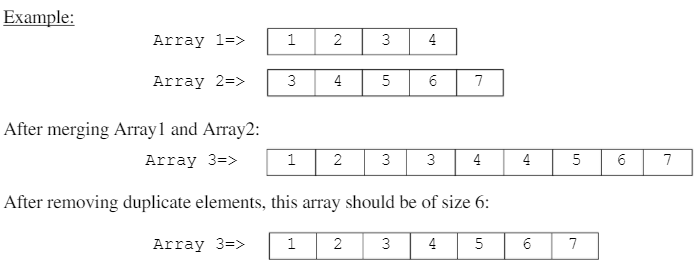
**Lab Task**

1. Write a function **resize()** that takes as arguments: a pointer pointing to the array of integers, its size, and a new\_size. new\_size can be any number greater than 0. This function should change the size of the array. If the new size is greater than the previous one, then insert zeroes in new cells.



|  |
| --- |
| void resize( int \*&arr, int size,int new\_size) { //passing parameters  cout << "please enter size" << endl;  cin >> size; //taking size  cout<<"enter elements of array";  for (int i = 0; i < size; i++) { // taking elements of array  cin >> arr[i];  }  int \*arr2 = new int(new\_size); //allocating memory  cout << "enter new size" << endl; //taking new size  cin >> new\_size;  if (new\_size <= 0) { //condition that if new size is less than zero  cout << "you have enter an negative number, the number you enter must be positive" << endl;  } else {  if (new\_size < size) { //if new size is less than the size of array 1  cout <<"the copied array is"<<endl;  for (int i = 0; i < new\_size; i++) {  arr2[i] = arr[i]; //copying array 1  cout<< arr2[i]; //displaying it  }  } else {  cout<<"the new array is";  for (int i = 0; i < new\_size; i++) {  arr2[i] = arr[i]; //copying array elements  arr[size]=arr[new\_size]=0; //when the array size is equal to other size then we store the value zero at that place by default  cout << arr2[i]; //displaying  }  delete []arr; //deleting arrays  }   }  delete []arr2; } cout<<"SIR KINDLY RUN THE TASKS INDIVIDUALLY "<<endl;  output:  A picture containing text  Description automatically generated  Text  Description automatically generated with medium confidence |

2. Write a code that merges two arrays. Create two dynamic arrays of sizes size\_1 and size\_2 respectively. Take input in these arrays from the user. Now create a third array of size (size\_1+size\_2) and insert all the elements of both arrays in this array. Remove the duplicate elements from this array and resize the array to a smaller size.



|  |
| --- |
| void mergeArray() {  int size\_1;  int size\_2;  int size; //declaring arrays  int \*arr1 = new int[size\_1]; //allocating memory  int \*arr2 = new int[size\_2];  cout << "enter size of arr1= ";  cin >> size\_1; //taking size1  cout << "enter size of arr2= ";  cin >> size\_2; //taking size2  size = size\_1 + size\_2;  cout << "size =" << size << endl; //displaying size that is sum of 1 and 2  int \*arr3 = new int[size];  cout << "enter values of arr1 " << endl;  for (int i = 0; i < size\_1; i++) { //taking elements of array1  cin >> arr1[i];  }  cout << "enter values of arr2 " << endl; //taking elements of array2  for (int i = 0; i < size\_2; i++) {  cin >> arr2[i];  }  for (int i = 0; i <size\_1; i++) {  arr3[i] = arr1[i]; //copying the array 1 in third array  cout<<arr3[i];  }  int j=0; //initializing j=0  for(int i=size\_1;i<size;i++){  arr3[i]=arr2[j]; //copying the arr3 value in arr3 after the size of array1  j++; //iterating  cout<<arr3[i]; //displaying array 3  }   delete []arr2; //deleting array 2 and 1  delete []arr1; // cout<<"\nfor deleting the array elements"<<endl;  for(int i=0;i<size;i++){ //the loop until size  for(int j=i+1;j<size;j++){ //the loop from the next index until size  if(arr3[i]==arr3[j]){ //checking first 2 indexes  for(int k=j;k<size;k++){ //checking the next indexes  arr3[k]=arr3[k+1];  }  size--; //decrementing the size of array  j--; //decrementing the j till the size  }  }  }  cout<<"\nafter deleting elements = ";  for(int i=0;i<size;i++){  cout<<arr3[i]<<" "; //displaying the array3  }  delete []arr3; //deleting the array 3  } cout<<"SIR KINDLY RUN THE TASKS INDIVIDUALLY "<<endl;  output:  Text  Description automatically generated |

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