National University of Computer and Emerging Sciences



**Laboratory Manual**

*for*

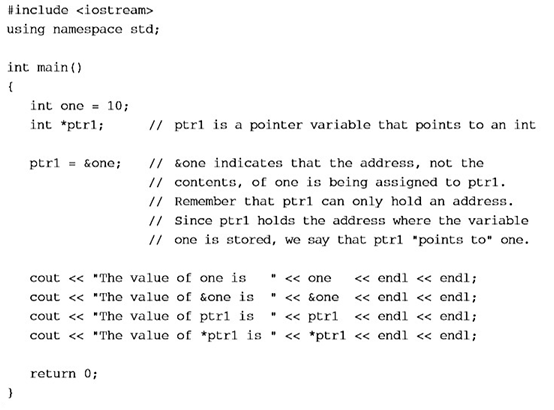
**Computer Programming**

|  |  |
| --- | --- |
| Course Instructor | Mr. Sarim Baig |
| Lab Instructor(s) | Zara Asif, Ahsan Rehman |
| Section | E |
| Semester | Spring 2017 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

Lab Manual # 3



Lab Manual # 3



**Objective:** Practice of dynamic allocation of 1 D array.

**Question 1:**

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.

Example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Array 1=> | 1 | 2 | 3 | 4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Array 2=> | 3 | 4 | 5 | 6 | 7 |

After merging Array1 and Array2:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Array 3=> | 1 | 2 | 3 | 3 | 4 | 4 | 5 | 6 | 7 |

After removing duplicate elements, this array should be of size 6:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Array 3=> | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

**Question 2:**

Write a function which returns true if the string parameter is a palindrome. A palindrome is any "word" which is the same forward and backward, eg, "radar", "noon", "20011002".The function should return false if the argument is not a palindrome. Assume the prototype is

bool isPalindrome(char \* a);

**Question 3:**

Write a program that:

* Asks the user how many names they wish to enter.
* Asks the user to enter each name.
* Calls a function to sort the names
* Prints the sorted list of names.

Hint: Use a dynamic array of string to hold the names.  
Hint: string supports comparing strings via the comparison operators <, and, >

Your output should match this:

How many names would you like to enter? 5

Enter name #1: Jason

Enter name #2: Mark

Enter name #3: Alex

Enter name #4: Chris

Enter name #5: John

Here is your sorted list:

Name #1: Alex

Name #2: Chris

Name #3: Jason

Name #4: John

Name #5: Mark

**Question 4:**

Consider following main function:

|  |
| --- |
| void main()  {  //For example, user enters National University.  ReverseSentence(q);  // Now input array should be changed to lanoitaNytisrevinU.  } |

Write the implementation of the function **void ReverseSentence(char\*)**. Assume that each sentence ends with a full stop. You should use the following function **ReverseWord()** to reverse each word of the whole sentence. You are not allowed to use any static array. You are only allowed to use dynamic char array.

|  |
| --- |
| void ReverseWord(char \*p, intlen)  {  char temp;  for(int i=0; i<len/2; i++)  {  temp=p[i];  p[i]=p[len-i-1];  p[len-i-1]=temp;  }  } |

“p” is a pointer pointing to the first location of *char* array and length is the number of characters in the array. For example, if p is pointing to “HELLO” then the length is 5. After calling this function, p is pointing to “OLLEH”.

Lab Manual # 3





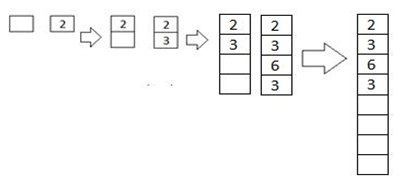
**POST LAB**

Make an array that dynamically grow and shrinks according to the volume of data in it.

Initially make a dynamic array of size 1.

Array grows dynamically (**don’t fix the size**) whenever user wants to add or delete element for that you need to resize the array on two conditions.

1. When no free places left then makes it double.



1. Whenever free places are greater than half of capacity. Resize the array and make it half. (Remove the half empty spaces).

Notice: By Deleting means delete the previously allocated array using **delete** Keyword..

**INSERTION RULE:** Moreover insertion will occur in ascending order and print out the place/index wherenew element is inserted (don't sort after the value is inserted, make sure you insert it in ascending order). If there is no space, make sure you resize array and then add the element.

e.g. we have array {2,5,7,8} and we want to insert 4 after insertion array look like this {2,4,5,7,8, ,-,-,-} 4 is inserted at 2nd place

**DELETION RULE:** Whereas in the case deletion, deletion can be done by inputting the elementpreviously inserted or index by user.

If user input an index delete that index and shift the next element to left e.g.

Before deletion array is {2,4,7,7,8,9,-,-} delete element at index 3 after deletion array is {2,4,7,8,9,-,-,- } Before deletion array is {2,4,7,8,9,-,-,-} delete element at index 2 after deletion array is {2,4,8,9, }

If user input an element delete all element from array, print all indexes from where element are deleted and shift array to left e.g.

Before deletion array is {2,4,7,7,8,9,-,-} delete 7 after deletion array is {2,4,8,9} 7 is deleted from 2nd and 3rd index

Note\* After deletion resize array if free places are greater than half of capacity