

**NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY**

**School of Electrical Engineering and Computer Sciences**

**Advanced Programming**

**(CS213)**

**Assignment#1**

**SUBMITTED TO:**

**Sir Fahad Satti**

**SUBMITTED BY:**

**Maria Hameed**

Reg. No. 05597

Class: BESE-4A

**Date of Submission: 01-03-2016**

Contents

[Introduction: 3](#_Toc444809991)

[**COW with reference counting:** 3](#_Toc444809992)

[**COW with reference counting:** 3](#_Toc444809993)

[**Owned Pointers:** 4](#_Toc444809994)

[**Copied pointers:** 4](#_Toc444809995)

[How to run program: 5](#_Toc444809996)

[Profiling: 5](#_Toc444809997)

[**COW with reference counting:** 5](#_Toc444809998)

[**COW with reference linkng:** 5](#_Toc444809999)

[**Copied pointers:** 6](#_Toc444810000)

[**Owned Pointers:** 6](#_Toc444810001)

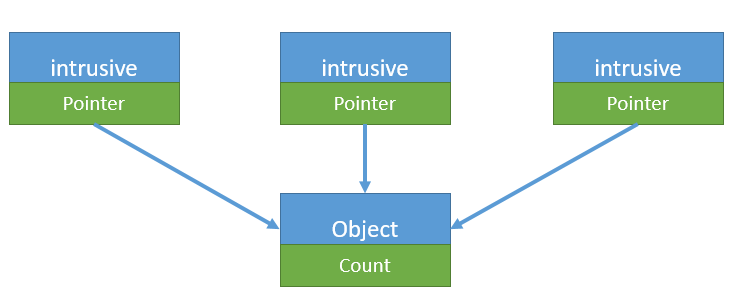
# Introduction:

We were provided with the basic structure for the StringBuffer Class.

In this assignment, implementation of StringBuffer, smart Pointers using COW with reference counting, COW with reference linking, owned pointers and copied pointers is done.

## **COW with reference counting:**

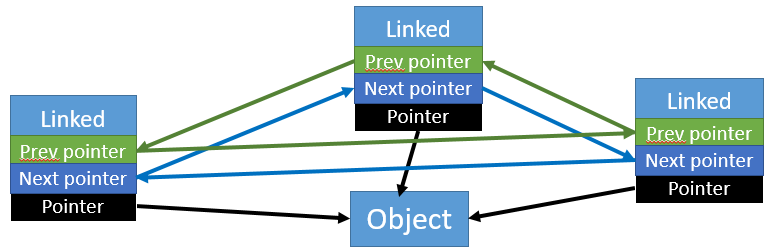
For smart Pointers using COW with reference counting, a **\_refcount** variable is added to the class StringBuffer. The basic idea is shown below



Whenever a new pointer starts pointing to an object, its **\_refcount** is incremented. On dereferencing, the counter is decremented. When the counter value decreases to zero, the object is also destroyed.

## **COW with reference counting:**

The basic idea is shown below:



Two pointers SP \*next, SP \*prev are added in the class “SP”. Whenever a new pointer start pointing to the object, its next and previous pointers are set. This forms a circular doubly linked list of all smart pointers that point to the same object. Whenever dereferencing occurs, the pointer is removed from the list. When the last element in the list is removed, the object is deleted as well.

## **Owned Pointers:**

When a new pointer starts pointing to an object, the old pointer pointing to is set to NULL. When the pointer pointing to the object is destroyed, the object is destroyed as well.

Object o1

Pointer p1

After assignment:

NULL

Pointer p1

Object o1

Pointer p2

## **Copied pointers:**

When a pointer is assigned to a new pointer, internally a new object is created and the new pointer starts pointing to that new object. In other words pointer p1= pointer p2 means that p1 and p2 point to their own separate copies.

Pointer p1

Pointer p2

Object

Object

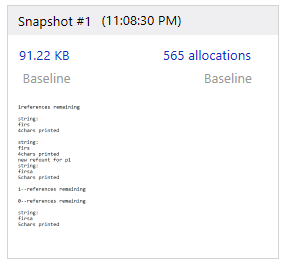
# How to run program:

Implementation of each of smart pointer is provided in a separate program.

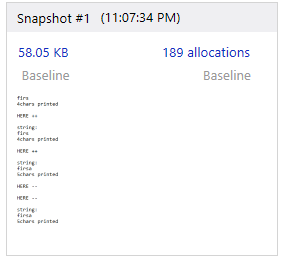
To check any of the implementation, just open the cpp file and run it.

# Profiling:

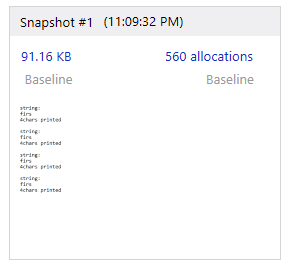
## **COW with reference counting:**



## **COW with reference linkng:**



## **Copied pointers:**



## **Owned Pointers:**

