University of Central Punjab
(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)
FACULTY OF INFORMATION TECHNOLOGY

Programming Fundamentals

Lab 11	
Topic	1D regrow and shrink

Objective	• POINTER HAZARDS
	  ○ Explaining Memory Leak
	  ○ Discuss heap overflow
	  ○ Explaining Dangling Pointer
	• Discuss Deep Copy and Shallow copy
	• Discuss issues with shallow copy.
	• Discuss the use of deep copy.
	• Auto-Growing of 1D arrays
	  ○ Need for auto-growing arrays.
	  ○ Technique for growing 1D arrays automatically.
	• Auto-Shrink of 1D arrays
	  ○ Need for auto-shrink arrays.
	  ○ Technique for shrinking 1D arrays automatically.

Lab Description:

This lab is basically designed for the pointer hazards, regrow and shrink. We will see the use of dep copy and shallow copy and different techniques of regrow and shrink.

Memory Leakage:

As we discussed earlier dynamic memory is allocate and deallocate by user. If we create a memory dynamically and accidently lost its base address so we are not able to delete that memory. A memory leak occurs when a piece (or pieces) of memory that was previously allocated by a programmer is not properly deallocated (free) by the programmer.

Heap overflow:

If we allocate new dynamic memory repeatedly without deallocating previous one it may cause heap overflow. Because at some point without deallocating the memory there will be no space left for new allocation.

Dangling Pointer:

If a pointer points illegally to a memory location it's called dangling pointer. Like a pointer holds the base address of a dynamic memory location which is not existing anymore.

Both issues are not acceptable in C++. One (**memory leakage**) is wasting memory that can be used by others. And another one (**dangling pointer**) is, illegally, pointing at the memory that is not reserved for it.

Shallow copy VS Deep copy:

Shallow Copy:

If we assign a value of a pointer to another pointer than both pointers hold the same value it means both pointers are pointing to same address. If you change anything using first pointer it effects the second pointer as well because both pointers share same memory.

Deep Copy:

Deep copy works differently as compare to shallow copy. In deep copy both pointers have their own memory location. One pointer holds the copy of the data of other pointer in its own memory location. If you change anything using first pointer it not effects the second pointer because both pointers have different memory.

Auto-Grow:

One of the benefit of dynamic memory is that we can change it according to our requirement. But we cannot directly change that memory. Once you allocate a memory you cannot modify it directly in order to grow it. You need to follow some steps in according to modify the allocated memory in order to grow it.

Steps for regrow memory:

1. Allocate new memory according to required size.
2. Copy the content of previously allocated memory into new allocated memory.
3. Delete previous allocated memory.
4. Return the base address of new allocated memory.

Example:

```
int* autoGrow(int*ptr, int size)
{
    int *temp = nullptr;
    temp = new int[size + 1];
    for (int i = 0; i < size; i++)
        *(temp + 1) = *(ptr + i);
    if (ptr != nullptr)
    {
        delete[]ptr;
        ptr = nullptr;
    }
    return temp;
}
```

Auto-Shrink:

One of the benefit of dynamic memory is that we can change it according to our requirement. But we cannot directly change that memory. Once you allocate a memory you cannot modify it directly in order to shrink it. You need to follow some steps in according to modify the allocated memory in order to shrink it.

Steps for regrow memory:

1. Allocate new memory according to required size.
2. Copy the content of previously allocated memory into new allocated memory.
3. Delete previous allocated memory.
4. Return the base address of new allocated memory.

Example:

```
int* autoShrink(int*ptr, int occ)
{
    int *temp = nullptr;
    temp = new int[occ];
    for (int i = 0; i < occ; i++)
        *(temp + 1) = *(ptr + i);
    if (ptr != nullptr)
    {
        delete[]ptr;
        ptr = nullptr;
    }
    return temp;
}
```

Lab Tasks

Registration problem:

Different students enroll in different courses. Each course have multiple sections. Some time we need to perform different operation on these kind of data. Which may modify the data. Record of each section is stored in a file with total student in each section.

Design a system which perform following operations:

- Enroll a student in a section.
- Remove a student from a section.
- Update the record of the student.
- Transfer a student from one section to another section.
- Identify those sections which have capacity to enroll more students.
- Identify those section which have no capacity to enroll more students.