# Department of Computing

# School of Electrical Engineering and Computer Science

**CS-250: Data Structure and Algorithms**

**Class: BEEE 1****2 (G1+G2)**

**Lab 2:****Dynamic Memory Allocation (I)**

**Date: 5th February, 2024**

**Time: 10:00 am – 12:50 am**

# Lab Engineer: Anum Asif

# Lab 2: Dynamic Memory Allocation (I)

**Introduction**

This lab is about dynamic memory allocation. Dynamic memory allocation in C/C++ refers to performing memory allocation manually by programmer. Dynamically allocated memory is allocated on **Heap** and non-static and local variables get memory allocated on **Stack**

**Objectives**

This lab will revise the old concepts taught to the students in the previous semesters.

**Tools/Software Requirement**

Visual Studio C++

**Description : Static vs Dynamic Arrays**

Consider two variants of declaring arrays below. Memory for the first variant gets allocated on the Stack. The lifetime of an array created using the method A depends on its scope. If it is defined globally, its life is equal to the lifetime of the application. If it is declared in a function, memory for it gets allocated on the stack when the function gets called. It gets deallocated when the function call terminates. All the data related to the function call including the array gets removed from the stack. On the other hand, memory for the array created using new operator gets allocated on the heap at runtime. The lifetime of such an array is at max equal to the execution time of the application. If the array is no more required, the memory allocated for it can be freed using **delete []** command.

**Method A:**

const int size=5;

int x[size];

for (int i = 0; i < size; i++)

{

//cout << "x[" << i << "] = ";

x[i] = i + 1;

}

**Method B**

int size; // Note that size variable is const in variant A whereas it isn’t in variant B. Find out the logic behind it.

cout << "Enter size of array: ";

cin >> size;

int \*x = new int[size];

for (int i = 0; i < size; i++)

{

//cout << "x[" << i << "] = ";

x[i] = i + 1;

}

**Lab Tasks**

**Task 1**

Assume that the following variable declaration has already been made:

char\* oddOrEven = " Never odd or even ";

Write a single statement to accomplish each of the following tasks (assuming for each one that the previous ones have already been run). Make sure you understand what happens in each of them.

1. Create a pointer to a char value named nthCharPtr pointing to the 6th character of oddOrEven (remember that the first item has index 0). Use the indexing operator.
2. Using pointer arithmetic, update nthCharPtr to point to the 4th character in oddOrEven.
3. Print the value currently pointed to by nthCharPtr.
4. Create a new pointer to a pointer (a char \*\*) named pointerPtr that points to nthCharPtr.

**Task 2**

Rewrite this program using pointers in place of arrays (use dynamic memory allocation operators new and delete). The syntax is int \*salArray= new int[size]; and for deletion delete [] salArray.

include<iostream>

using namespace std;

int main(void)

{

int salary[20];

inti;

for (i=0;i<20;++i)

{

cout<<"Enter Salary: ";

cin>>salary[i];

}

for (i=0;i<20;++i)

salary[i]=salary[i]+salary[i]/(i+1);

return 0;

}

**Task 3**

Write a C++ Program to get GPA of n number of students and display it.

// where n is the number of students entered by the user

Note: Dynamically store GPA’s

**Deliverables**

Compile a single word document by filling in the solution part and submit this Word file on LMS. The name of word document should follow this format. i.e. **YourFullName(reg)\_Lab#.** You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on the LMS.

**Note:** Students are required to upload the lab on LMS before deadline.