National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

School of Electrical Engineering and Computer Science

CS-250: Data Structure and Algorithms

Class: BS-EE 13

Lab 1: Pointers in C++

Date: 29th January, 2025

Time: 02:00 PM – 04:50 PM

Instructor: Ms.Ayesha Sarwer

Lab Engineer: Areeba Rameen

Lab 1: Pointers in C++

Introduction

This lab is about the pointers. In C++, a pointer refers to a variable that holds the address of another variable. Like regular variables, pointers have a data type. For example, a pointer of type integer can hold the address of a variable of type integer. A pointer of character type can hold the address of a variable of character type.

Objectives

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

Tools/Software Requirement

Visual Studio C++ / Github Code Spaces

Description

Pointers are used to point towards a particular memory address. In this lab we will use the pointers and perform task with the help of them.

Lab Tasks

Task 1

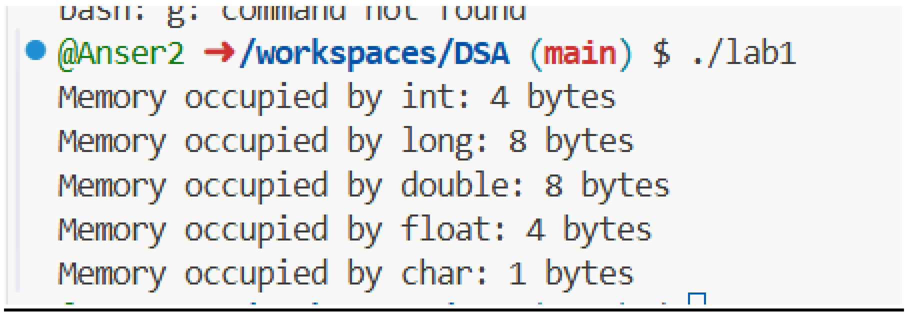
Write code to find the memory in bytes occupied by int, long, double, float and char.

```
#include <iostream>

int main() {   // using the sizeof method

   std::cout << "Memory occupied by int: " <<sizeof(int) << " bytes" << std::endl;
   std::cout << "Memory occupied by long: " << sizeof(long) << " bytes" << std::endl;
   std::cout << "Memory occupied by double: " << sizeof(double) << " bytes" << std::endl;
   std::cout << "Memory occupied by float: " << sizeof(float) << " bytes" << std::endl;
   std::cout << "Memory occupied by char: " << sizeof(char) << " bytes" << std::endl;

   return 0;
}
```



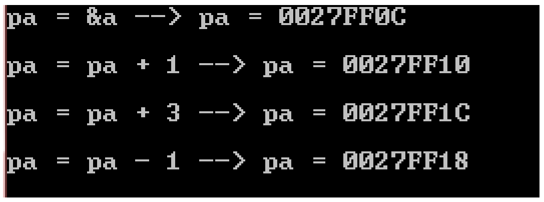
```
bash: g: command not found
@Anser2 ➜/workspaces/DSA (main) $ ./lab1
Memory occupied by int: 4 bytes
Memory occupied by long: 8 bytes
Memory occupied by double: 8 bytes
Memory occupied by float: 4 bytes
Memory occupied by char: 1 bytes
```

Task 2

Consider the following program and answer the questions.

```
void main()
{
int a, *pa;     // Statement 1
pa = &a;        // Statement 2
cout<<"pa = &a --> pa = "<<pa<<endl<<endl;
pa = pa + 1;    // Statement 3
cout<<"pa = pa + 1 --> pa = "<<pa<<endl<<endl;
pa = pa + 3;    // Statement 4
cout<<"pa = pa + 3 --> pa = "<<pa<<endl<<endl;
pa = pa - 1;    // Statement 5
cout<<"pa = pa - 1 --> pa = "<<pa<<endl<<endl;
}
```

Output:



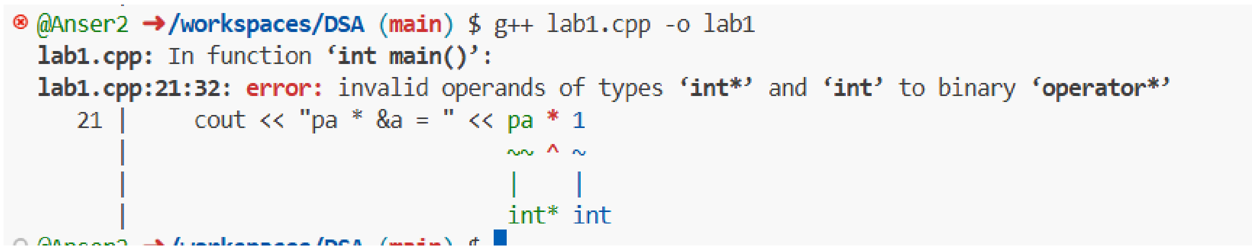
```
pa = &a --> pa = 0027FF0C

pa = pa + 1 --> pa = 0027FF10

pa = pa + 3 --> pa = 0027FF1C

pa = pa - 1 --> pa = 0027FF18
```

1) Why does the memory address stored in pointer “pa” vary by 4?
   ANS: ***It’s because we are adding an integer, which has a size of 4 bytes. Hence, the address keeps varying by 4.***
2) Will the address still vary by 4 if the data type of the above-mentioned code changed from “int” to “long”? Explain your answer.
   ANS*: **No, it will vary according to the digit will be added. For example, in case if a character is added it will vary by 1.***
3) If we try to multiply the address pointed to by “pa” what will happen? Is this logically or programmatically correct? Attach screen shot of the output you get when you try this multiplication.
   ANS: ***It’s not programmatically or logically correct. We can’t multiply pointer address with a pointer (invalid operand types).***



```
⊗ @Anser2 ➜ /workspaces/DSA (main) $ g++ lab1.cpp -o lab1
  lab1.cpp: In function ‘int main()’:
  lab1.cpp:21:32: error: invalid operands of types ‘int*’ and ‘int’ to binary ‘operator*’
     21 |     cout << "pa * &a = " << pa * 1
        |                             ~~ ^ ~
        |                             |    |
        |                             int* int
```

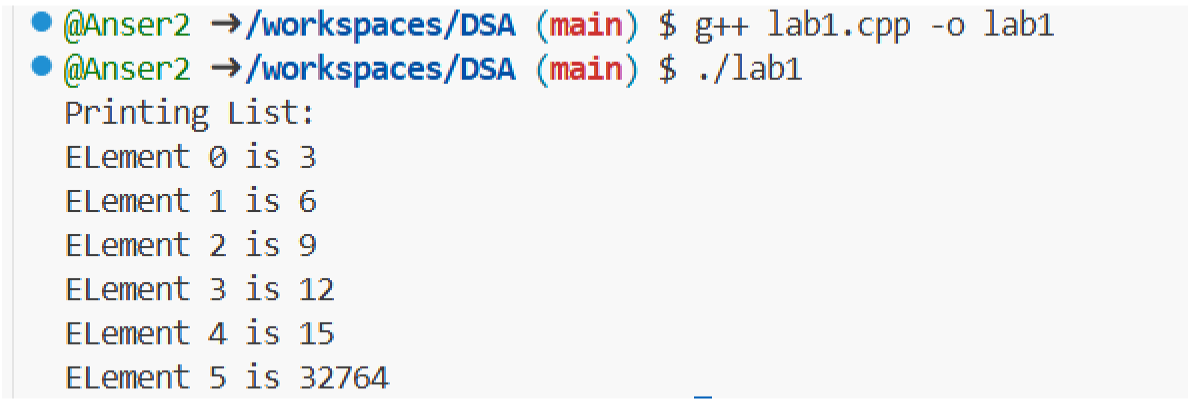
Task 3

int list[5]={3,6,9,12,15};
int *pArr= list;

Your task is to write a piece of code that prints all values stored in the array **list** using only pointer variable pArr. Do not use the conventional way of printing values by numbering indexes.

```
#include <iostream>
using namespace std;
int main() {

    int list[5]={3,6,9,12,15};
    int *pArr= list;
    cout << "Printing List: " << endl;   // Looping and incrementing
    for(int i=0; i<=5; i++){
            cout << "ELement "<< i <<" is " << *(pArr + i) << endl;
    }
    return 0;
}
```



```
● @Anser2 →/workspaces/DSA (main) $ g++ lab1.cpp -o lab1
● @Anser2 →/workspaces/DSA (main) $ ./lab1
  Printing List:
  ELement 0 is 3
  ELement 1 is 6
  ELement 2 is 9
  ELement 3 is 12
  ELement 4 is 15
  ELement 5 is 32764
```

Task 4

Write output of the following C++ codes in your document without executing it.

Example code a)

```
int x[4] = {0,4,6,9};
int *p, a=3;
p=x;
(*p)++;   // 1
cout<<*p<<endl;
cout<<*(p+1)<<endl;
p++;
*p=*p+a;
cout<<*p<<endl;
p=p+2; //What is happening here?
cout<<*p<<endl;
```

Output:

1 4 7 9

Example code b)

```
int a, *p, *q;
int arr[4]= {0};

p=arr;
q=p;

*p=4;

for(int i=0; i<3; i++){
            a=*p;
            p++;
            *p=(a+i);
}
for (int j=0; j<4; j++){
            cout<<*q<<" ";
            q++;
}
```

Output:

4 , 4, 5, 7

Task 5

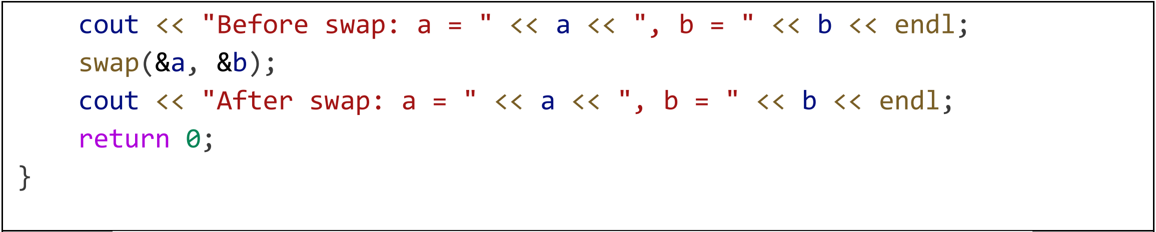
```
int a=5, b=10;
int *pa=&a; //pa and pb are pointer variables of type int.
int *pb=&b;

int **ppa=&pa; //ppa and ppb are called double pointers or pointers-to-pointers.
int **ppb=&pb;
```

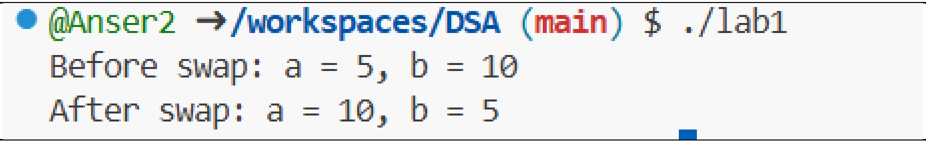
a) Write code of a function that swaps values of variables a and b. Input to the function should be the address of both the variables.

```
void swap(int *x, int *y) {      // Using a temp variable to swap
    int temp = *x;
    *x = *y;
    *y = temp;
}

int main() {
    int a = 5, b = 10;
```



```
    cout << "Before swap: a = " << a << ", b = " << b << endl;
    swap(&a, &b);
    cout << "After swap: a = " << a << ", b = " << b << endl;
    return 0;
}
```



```
● @Anser2 →/workspaces/DSA (main) $ ./lab1
  Before swap: a = 5, b = 10
  After swap: a = 10, b = 5
```

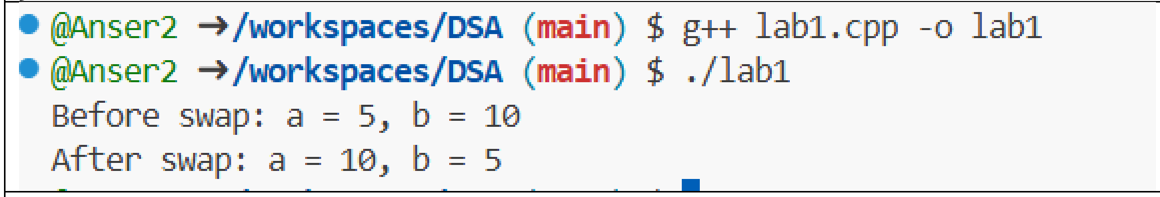
b) Write code of a function that swaps values of the variables a and b using pointer-to-pointer variables ppa and ppb.

```
#include <iostream>
using namespace std;

void swapUsingPointerToPointer(int **x, int **y) {
   int temp = **x;          // Using a temp variable to swap
   **x = **y;
   **y = temp;
}

int main() {
   int a = 5, b = 10;
   int *pa = &a;
   int *pb = &b;
   int **ppa = &pa;
   int **ppb = &pb;

   cout << "Before swap: a = " << a << ", b = " << b << endl;
   swapUsingPointerToPointer(ppa, ppb);
   cout << "After swap: a = " << a << ", b = " << b << endl;
   return 0;
}
```



```
● @Anser2 →/workspaces/DSA (main) $ g++ lab1.cpp -o lab1
● @Anser2 →/workspaces/DSA (main) $ ./lab1
  Before swap: a = 5, b = 10
  After swap: a = 10, b = 5
```

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 a 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.

Use proper indentation and comments. Lack of comments and indentation will result in deduction of marks.

Lab Rubrics

Assessment	**Doesn't meet Expectation (1-2)**	**Meets Expectation (3-4)**	**Exceeds Expectation (5)**	**Marks**
Software Problem Realization (CLO2 – PLO2)	The Student is unable to understand and **outline** the problem and doesn't use the relevant method to solve it.	The student requires some guidance to completely comprehend the problem and to **differentiate** the data structure and algorithm comprehensively.	The student fully understands the given problem, is able to **analyze** the relevant method to solve it, and develops a detailed program flow.	
Software Tool Usage (CLO3 – PLO5)	The student has no idea on how to use the basic tools of the software. The codes have syntax errors, and parts of the codes are missing. Also, they are unable to **imitate** the required output	The student has a limited command on the basic tools of the software and **operated it under supervision**. The codes are correct in terms of their syntax, however, the program output is not always correct.	The student has full command on various tools available in the software. Furthermore, his/her coding is complete and functional, and the program output is correct. Moreover, they can easily	

			manipulate the code to design a particular solution	
Ethics and Adherence to Laboratory Safety Rules (CLO4 - PLO 8)	The student does not **behave according to** the professional ethics by following ethical norms applicable to the software industry such as acknowledgement while using publicly available data/ code. Disturbs the lab environment, doesn't take care of safety measures, and/or isn't punctual.	The student partially demonstrate their commitment to professional ethics by following ethical norms applicable to the software industry such as referencing and acknowledgement while using publicly available data/ code. **Exhibits** better behavior, works by taking into account the safety measures, and is punctual.	The student clearly **express** the commitment to professional ethics by following ethical norms applicable to the software industry such as referencing and acknowledgement while using publicly available data/ code. Encourages others to maintain lab decorum, and alerts them to follow safety measures.	