**Name: Ahmad Amjad Mughal**

**Reg No: 121672**

**Class: BSCS-6C**

**Lab 1**

**Data Structures and Algorithms Lab 1**

**Task 1:**

**Main Purpose:**

In first we have to rewrite the program using pointers to understand the use of pointers and what its purpose to use.

//Ahmad Amjad Mughal

//BSCS-6C Roll No 121672

#include<iostream> //library

using namespace std;

int main(void) //main funtion

{

int \*salary; //pointer declaration

salary = new int[20]; //Assign the memory location of int array to slaray pointer

int i;

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

{

cout << "Enter Salary: "; //prompt

cin >> salary[i];

}

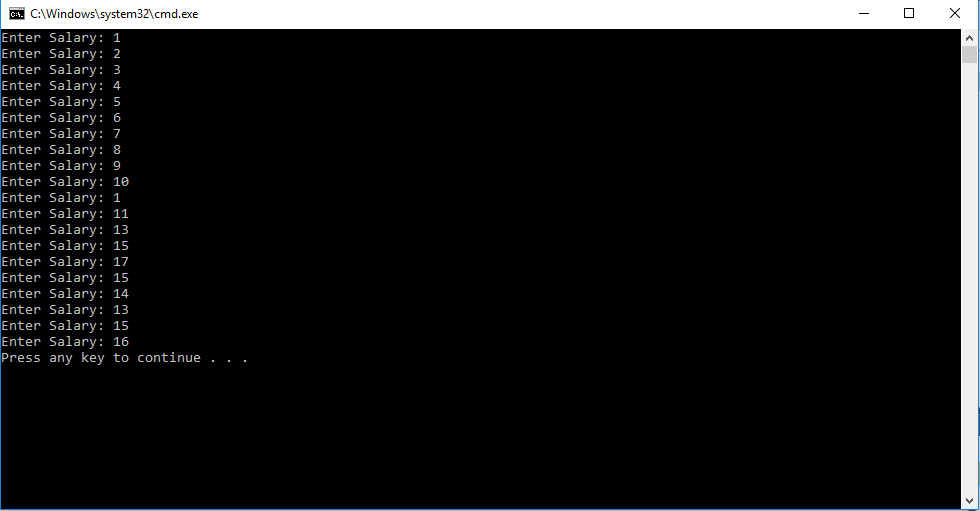
for (i = 0; i<20; ++i)

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

return 0; //return successfullty

}

**Output:**



**Task 2:**

**Main Purpose:**

The main purpose and that point to understand in this lab is function calls, use of pointers in calling function that hold the address of primitive type variable values. The difference between stack memory allocation and dynamic memory allocation. For stack there must be a garbage address (hidden) for that variable when we go out of that scope or we call function in main function.

For case of dynamic memory allocation, there is no garbage value.

//Ahmad Amjad Mughal

//Reg No 121672

//BSCS 6C

#include<iostream> //library inputoutputstream

using namespace std;

void analyze\_pointer(int \*ptr) //function that uses pointer ptr

{

cout << endl << "The Adress of this int is:"; //for printing adress of integer

cout << ptr<<endl;

cout << "Value of int using pointer is:"; //for printing value of integer

cout << \*ptr<<endl;

}

int \*function1(){ //function of data type pointer for stack memory allocation

int fun = 6;

return &fun; //return adress

}

int \*function2(){ //function of data type pointer for dynamic memory allocation

int \*pointer = new int;

\*pointer = 6;

return pointer; //reurn pointer that hold adress of integer

}

int main() //Main function

{

cout << "Part #2: Memory on stack..."; //prompt

int stack = 1;

analyze\_pointer(&stack); //function call that prints inter value

analyze\_pointer(function1()); //calling function that allocates stack memory

cout << "Memory on heap..." << endl;

int \*point = new int; //code for dynamic memory

int heap = 3;

point = &heap;

analyze\_pointer(point); //function call

analyze\_pointer(function2()); //calling functions that allocates dynamic memory

return 0;

}

**Output:**

