**Task no.1**

#include<iostream> //header files

using namespace std;

int main(void)

{//scope begins

int \*salary; //int pointer of salary

salary = new int[20]; //heap memory allocation to salary

for (int i = 0; i<20; ++i) //for loop

{

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

cin >> \*(salary + i); //taking input from user

}

for (int i = 0; i<20; ++i) //for loop

\*(salary + i) = \*(salary + i) + \*(salary + i) / (i + 1); //arithmetic calculations using pointer

delete[]salary; //delete heap memory in salary

return 0; //return 0 to main function

} //scope ends

**Task no.2**

**Part 1**

void analyze\_pointer(int \*ptr){ //function of analyze\_pointer using argument int pointer

cout << "Given address " << ptr << endl;//displaying address

cout << "Integer lies on given address " << \*ptr << endl;//displaying integer

}

**Part 2**

#include<iostream>//libraries

using namespace std;

void analyze\_pointer(int \*ptr){//previous function

cout << "Given address " << ptr << endl;

cout << "Integer lies on given address " << \*ptr << endl;

}

int main(void){

int iValue;//stack integer iValue

iValue = 12;//assign 12 to iValue

analyze\_pointer(&iValue);//Calling function using argument iValue by reference

int \*iiValue = new int;//heap memory to iiValue

\*iiValue = 12;//assigning value to pointer iiValue

analyze\_pointer(iiValue);//Calling function using argument iiValue

delete iiValue;//delete heap value

return 0;

}//scope ends

**Reason 1:** Both method are correct .The difference is static and dynamic allocation of memory.

**Part 3**

#include<iostream>//libraries

using namespace std;

void analyze\_pointer(int \*ptr){//previous function

cout << "Given address " << ptr << endl;

cout << "Integer lies on given address " << \*ptr << endl;

}

int\* int\_pointer1(){//int\_pointer1 function

int \*nptr = new int;//heap method

\*nptr = 13;

return nptr;

}

int\* int\_pointer2(){//int\_pointer2 function

int nptr;//stack method

nptr = 123;

return &nptr;

}

int main(void){

int iValue;//previous

iValue = 12;

analyze\_pointer(&iValue);

int \*iiValue = new int;//previous

\*iiValue = 12;

analyze\_pointer(iiValue);

delete iiValue;

/\*analyze\_pointer(int\_pointer1());\*/ //calling function in function

analyze\_pointer(int\_pointer2());//calling function in function

analyze\_pointer(new int);//calling function using argument heap of integer

return 0;

}//scope ends

**Reason2:** The memory leak in **int\_pointer1** case because we didn’t delete the pointer. If we do so then garbage value will return by function **int\_pointer1.** Garbage value in **int\_pointer2** case because we leave the scope, due to which garbage value assigned to the return pointer of **int\_pointer2.**

**Reason3:**

1. int\_pointer1 function causes memory leak.
2. new int causes memory leak.

**Task no.3**

#include<iostream>//libraries

#include<string>

using namespace std;

struct Area{//stucture namely area

private://private data

string units;

float area\_Value;

public://public data

string get\_unit(){//setters getters

return units;

}

void set\_unit(string a){

units = a;

}

float get\_area(){

return area\_Value;

}

void set\_area(float a){

area\_Value = a;

}

float half\_area(){//half area function

return area\_Value / 2;

}

float quarter\_area(){//quarter area function

return area\_Value / 4;

}

};

int main(){

Area\* x = new Area;//heap memory allocation

string a;

float b;

cout << "The unit of Area is ";

cin >> a;

cout << endl;

cout << "The Area is ";

cin >> b;

cout << endl;

x->set\_unit(a);//calling member function of x

x->set\_area(b);//calling member function of x

cout << "The half area is " << x->half\_area() << endl;//display

cout << "The quarter area is " << x->quarter\_area() << endl;//display

delete x;//delete heap memory

return 0;

}