20k-0368
Hammad Shakeel

ASSIGNMENT No 03

Q No 1

(a) ERRORS : We are deleting the same piece of memory two once in a destructor and other by called the member function. Deleting the same piece of memory twice can cause your program to crash or result in an "unusual behaviour".

Error - Free Code :

```
#include <iostream>
using namespace std;
class A {
        int  *x ;
    public :
        A() {
            x = new int ;
        }
        ~A() {
            delete x ;
        }
};

int main() {
    A a ;
    // a.memory-free() ;   ● No need for this
}
```

We just removed the member function. Since the memory will be freed when the destructor will be called.

Q no 1 (b)

ERROR : Static data members of the class should be defined outside the class (globally) with the name of respective class and scope resolution (::) operator. But in this program we create a static data memeber i.e. y, but we didn't declare it outside the class.

Error - free code :

```
#include <iostream>
using namespace std;
class ABC {
        int x ;
        static int y ;
    public :
        ABC () {
            x = 0 ;
            y = 0 ;
        }
    ABC (int a, int b) {
            x = a ;
            y = b ;
        }
  };
  int ABC :: y = 0 ;            // declaration of static member

  int main () {

        ABC a ;

        ABC b (1, 2);

  }
```

Q no 2

(a)

Ans The copy-constructor (user-defined) is required only if an object has pointers or any runtime allocation of the resource like file handle, etc. Default constructor does only shallow copy. For the deep copy we need to define a user-defined copy constructor. In user-defined copy constructor we make sure that pointers (or reference) of copied ~~constructor~~ object point to new memory location.

(b)

Ans Yes, it is necessary in some cases. The purpose of setter is to set the values of data member of that object at anytime. But the constructor can initialize the values only once when the object is created. So if the user want to change the values in the middle of the program so, the constructor can not be called, so we have to have a setter function to set value at any time.

(c)

Ans We can either initialize a const data member ~~[illegible]~~ at the time of its declaration or can use constructor initialization list either to set it or even change it only once. below is the code snippet.

```
class Constant {
        int x = 0
    Const int y = 0;        //set the value here or in the constructor list
    public :
        Constant ( ) : x(2), y(3) {}        // it can update the value
                                            // of constant but after this
                                            // statement it would be
                                            // constant throughout the
                                            // program
};
```

(d)

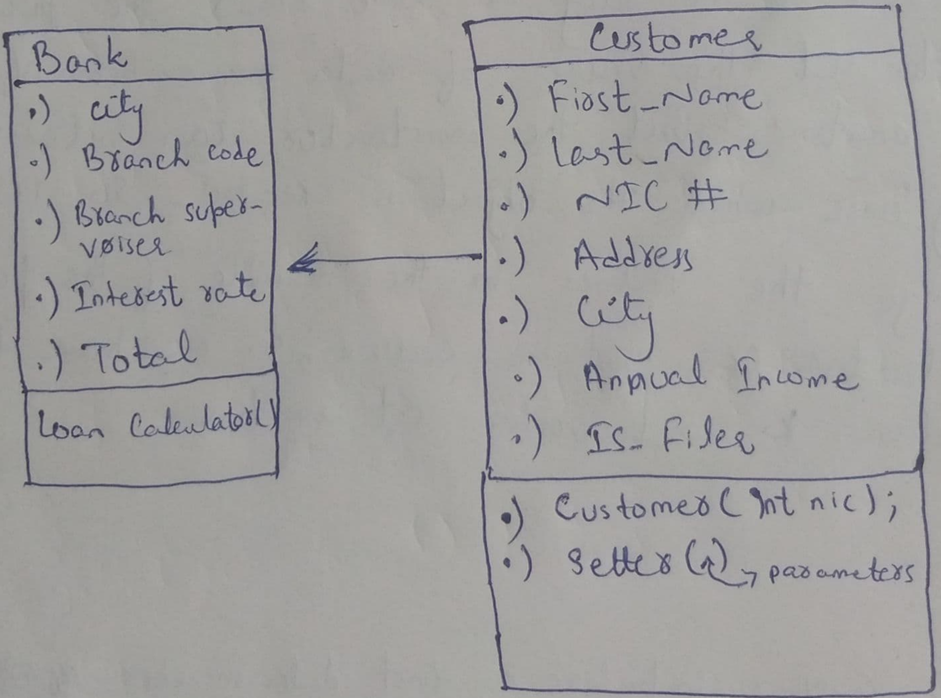
Ans. Since the constructors are special type of member function, they are not called like other functions, so they don't return values like other function. Their purpose is to initialize or construct the data members for the object of that class.

—

Q No 3

a) (i) Bank
(ii) Customer.

(b)



(c) class Customer{
private :
string first_name, last_name, address, city ;
const long int NIC ;
Long int income ;
bool ~~[illegible]~~ Is_filer ;

```
public :
    Customer ( long int NIC) : NIC (Nic) { }
    void setter ( string ~~name~~ first_Name, string Last_Name, long int income,
                  string address, string city, bool is_filer ){
        this-> first_name  = first_Name ;
        this-> last_name   = last_ Name ;
        this-> income      = income ;
        this-> address     = address ;
        this-> city        = city ;
        this-> Is_filer    = is_filer ;
    }
};
```

(d)

```
Customer ( string first_Name, string l_name, long int NIC, long int a_income,
           string address, string city , bool is_filer){
    first_name      = first_Name ;
    last_name       = l_name ;
    this-> NIC      = NIC ;   income = a_income ;
  this-> address    = address ;   this-> city = city ;
    Isfiler         = is_filer ; }
```

(e)

```
Bank  branch1 (0.5) , branch2 (0.7) , branch3 (0.1) ;
```

(f)

```
void IssueLoan (Customer c , float r_loan){
    if ( r_loan < c.get_income ) {
        if ( city == c. get_city ) {
              cout << "we have confirmed your loan issuance" ;
        }
        else {
              cout << "Opps! sorry" ; }
    } }
```

/* We assumed that we have get_city and get income as member functions of class customer so that we can access private data members of class customer and also Issue loan is the member function of class Bank. */

```
(g) void   LoanCalculator ( Bank branch , float loan)
    {   int ~~total~~ total ;
        total = ~~[illegible]~~ loan + ( loan * branch.getinterest() ) ;

        cout << "Total payable amount is " << total ;
    }
```
// we assume that we a member function getinterest in
// Bank class to access the interest data member .

```
(h)   Class Customer {
        private :
          string first-name , last-name , address ;
          Const long int nic ;
          bool Is-filer ;
        public :
          Customer ( long int nic ) : nic(nic) {}
          Customer (Const Customer &customer , long int nic ) : nic(nic) {
             first-name  = customer.first-name ;
             last-name   = customer.last-name ;
             address     = customer. ~~last-name~~; address ;
             Is-filer    = customer . Is-filer ;
        }} ;
```

(i) If we make our ~~LoanCalculator~~ loan Issued function static and ~~[illegible]~~ also the data member total loan static then whenever a customer would be given loan the value ~~[illegible]~~ of total loan would increase In this way we can get ~~[illegible]~~ the total amount of loan given to all the customer.

```
Static void loanIssued () { .... }

float Class-name :: total-loan = 0 ;
```