1	2	3	4	5
3	3	4.5	5	4

19.5/20

CHENNAI MATHEMATICAL INSTITUTE
https://www.cmi.ac.in

Mid Semester Examination, Jan–Apr 2024

Name: Aritra	Roll Number: MCS202304
Subject: PLC	Date:
Course & Year:	Total No. of Pages:

—— Begin here ——

1>

```
Public class Stack<[redacted] T> {
    private T[] data = new T[100];
    private int size;
    ...
    public boolean isEmpty() { ... }
    public void push<S extends T>(S o){...}
    public T pop() {...}
}
```

May not be allowed, but OK

OK

(3)

) are ≤ 10" 2) public class Truck extends Vehicle {
= 10"

than.

in

```
public boolean equals (Object o){
    if ( o instanceof Truck){
        if ((Truck) o. wheels() == this . wheels()){
            return true;
    else if ( o instanceof Car) {
        if ((Car) o. doors() == this .doors() ){
            return true
        }
    }
    else {
        return false;
    }
}
}
Public Class Car extends Vehicle {
    public boolean equals ( Object o){
        if (o instanceof Car){
            if ((car)o. doors() == this .doors() ){
                return true
            }
        }
        else if (o instanceof Truck) {
            if ((Truck)o. doors() == this .doors) ){
                return true
            }
        }
        return false
    }
}
```

3) (b)

(3)

5) (b)

```
fn fib(n: i32) -> i32 {
    let mut res = if n<0 {
                      0
                  } else if n==1 {
                      1
                  } else {
                      0
                  };
    let mut i = 2;
    let mut fib1 = 0;
    let mut fib2 = 1;
    while (i ≤ n) {
        res = fib1 + fib2;
        fib1 = fib2
        fib2 = res
        i = i+1
    }
    return res;
}
```

// turning res mut
So that we can modify it later

// remove let, and let the outer scope res be updated.

4

(a) fib(3) returns 0 ✓

because, let res = if n<0 {0}
else if n==1 {1}
else {0};
gets value 0 on input 3

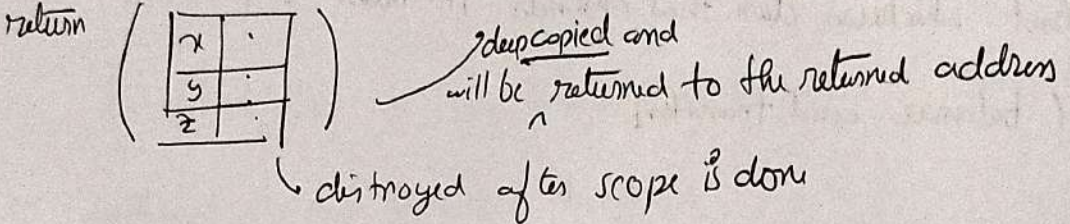
now, though the while loop will run 2 times
res is again defined inside scope of while.
So, value of that res will be lost and outer-scope res will not be updated. So will return 0.

Struct requires constant memory and therefore stored in stack.

(4) makepoint 1 : (works)

makepoint 1 creates a point object and returns its value [redacted] to the np variable in main().

then main prints the value of its fields.



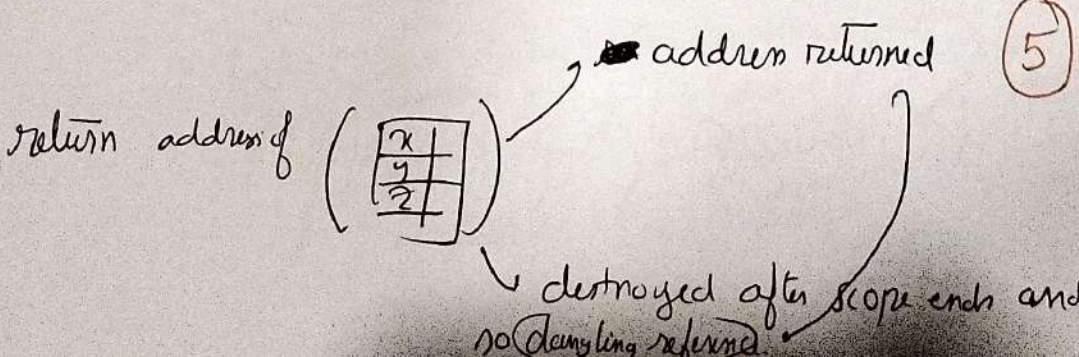
makepoint 2 : does not work.

makepoint 2 creates a point object, store it at p variable and tries to return the reference of that p variable.

But as soon as the scope of function (which is same as p variable) ends, p is erased from the stack (lifetime of p ends).

So, the reference returned is now dangling reference.

So, it gives compile time error.



5

③ the main problem is we can't give the whole bank account object to the user as it may have many secure functions that might be public

To overcome this, we first create a query interface so that whichever class that extends it must implement those function (balance and transfer)

Now, if we create a class that implements this interface for our object of them will have ~~only~~ two function

and when a user log in, upon checking credentials he will be given an object of this implemented class which will only allow users to have very ~~limit~~ limited access. and will be logged out upon using quota

(5)

public interface QueryInterface {

public double getbalance_fn();

public booban transfer_fn();

} // defining a query interface so that object of classes that implement it must have two functions

Public . BankAccount :

public double get_balance(int accno){

// return content of accno

}

public booban tranfer.(int src, int tgt, double d){

// transfer and return true if successful.

}

public class QuerySystem implements QueryInterface{

private int count = 1

public double getBalance_fn(){

get

}

(inside bank account class)

```
private
public class QuerySystem implements QueryInterface {
private
    int accno;
    private int quota;

    private QuerySystem(int _accno)
    {
        accno = _accno;
        quota = 1;
    }

    public double get_balance_fn(){
        if quota == 0, logout()
        quota--;
        return getbalance(accno).

    }

    public double transfer_fn(int, src, int tgt){
        if quota == 0, logout
        quota--;
        return transfer(accno, src, tgt)

    }
}
public QuerySystem login(int accno, string
                         , string P)

    // check credentials

    // if ok
        return new QuerySystem

    else return null

}
```

(4.5)

QueryInterface

queryobject

on login usr will only be able to query a single time and then will be logged out.