Programming Language Concepts

Mid-Semester Examination, II Semester, 2023–2024

Date : 26 February, 2024
Duration : 2 hours

Marks : 20
Weightage : 20%

- *Answer all questions.*
- *In any question that asks for code, provide Java or Rust style pseudocode, as appropriate. Syntax errors will be ignored, provided the code is conceptually correct.*
- *Supply explanations for any code you write, ideally as annotations alongside the code.*

---

1. Here is a skeleton definition in Java of a class `Stack` that implements a generic stack as an array of `Object`:.

   Rewrite this skeleton using type variables so that each instance of `Stack` can store any value that is a subtype of the concrete type supplied when instantiating `Stack`.

   (3 marks)

```
public class Stack {
  private Object[] data = new Object[100];
  private int size;
  ...
  public boolean isEmpty(){...}
  public Object pop(){...}
  public void push(Object o){...}
}
```

2. Consider the class definitions in Java given to the right.

   Explain how to override `Object.equals()` in classes `Truck` and `Car` so that two `Vehicle` objects are deemed equal if

   - Both are of type `Truck` and the number of wheels match.
   - Both are of type `Car` and the number of doors match.
   - One is of type `Car` and one is of type `Truck` and the number of doors match.

   (3 marks)

```
public abstract class Vehicle{
  public abstract int doors();
  public abstract int wheels();
}

public class Truck extends Vehicle{
  private boolean heavy;
  public Truck(boolean isheavy) {heavy = isheavy;}
  public int doors() {return 2;}
  public int wheels() {if (heavy) {return 6;}
                       else {return 4;}}
}

public class Car extends Vehicle{
  private boolean sporty;
  public Car(boolean issporty) {sporty = issporty;}
  public int doors() {if (sporty) {return 2;}
                      else {return 4;}}
  public int wheels() {return 4;}
}
```

3. All the savings accounts of a bank that supports online banking are stored in a single object of class `BankAccount`. This class supports functions for balance enquiries and transfers between accounts in the bank. The corresponding functions have the following signatures:

```
public double get_balance(int accno);
  // return contents of account accno
public boolean transfer(int src, int tgt, double d);
  // transfer amount d from account src to account tgt
  // return true iff successful
```

To access an account, a user must login with a valid username and password. After logging in, the user can perform *one* balance enquiry or transfer. To invoke another transaction, the user must log in again. There is no restriction on a user logging in multiple times in parallel. Each successful login in parallel can separately perform one transaction.

Logging into the account is implemented by a function of the form

```
public ... login(int accno, String u, String p);
// log in as user u with password p to access account accno
```

Suggest how we can implement the `login` function to provide the required limited access to `BankAccount`. Describe your design in terms of Java-like pseudocode. Give details of data definitions and function interfaces. For function bodies, you can write comments instead of detailed pseudocode to explain the features of your design.

(5 marks)

4. Two Rust functions, `makepoint1` and `makepoint2`, are defined below. Explain if one or both of them work/do not work as expected. (Give full explanations in terms of moving, borrowing, references, lifetimes, ...)

```
struct Point { x : i32, y : i32, z : i32 }

fn makepoint1(a:i32,b:i32) -> Point {
    let p = Point {x : a, y: b, z: 0};
    return p;
}

fn main1(){
    let np = makepoint1(5,7);
    println!("({},{},{}", np.x,np.y,np.z);
}
```

```
fn makepoint2(a:i32,b:i32) -> &Point {
    let p = Point {x : a, y: b, z: 0};
    return &p;
}

fn main2(){
    let np = makepoint2(5,7);
    println!("({},{},{}", np.x,np.y,np.z);
}
```

(5 marks)

5. Consider the following Rust function.

```
fn fib(n:i32) -> i32{
    let 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 {
        let res = fib1 + fib2;
        fib1 = fib2;
        fib2 = res;
        i = i+1;
    }
    return(res);
}
```

- Explain what `fib(3)` returns.
- How would you fix the function so that `fib(m)` returns the m^{th} number in the Fibonacci sequence?

(4 marks)