National University of Computer & Emerging Sciences, Karachi Campus
CS103 – Computer Programming – Spring 2015
Programming Assignment No. 3

Due Date: Sunday, April 19, 2015 [11:55 PM] **Total Points: 50**

The management of FAST-NU is planning to launch **"SahulatCard"** to facilitate their students in order to access many facilities and services both on and off campus. You will use it every day for food, photocopying, libraries, and computer labs.
The SahulatCard has multiple uses, including

Debit Card:	The SahulatCard is a debit card for the student's Flex Rupees account. Money can be added to the account at any time. Purchases paid for with Flex Rupees are subject to 17% GST imposed by Government.
Meal Plan Card:	If the student buys a meal plan for the term, then an initial amount of funds are put into the student's Meal Plan Funds account and the SahulatCard acts as a debit card for this account. Most meal plans offer discounts on all such purchases. Also, meal plan purchases are tax exempt (i.e., they are not subject to GST). If the Meal Plan Funds account runs out of funds, then food purchases are automatically charged to the Flex Rupees account, which means that no discount applies and there is GST.
Expired Meal Plan Card:	If there is money left in a student's Meal Plan Funds account at the end of a term, those funds are transferred to the student's Transfer Meal Plan Funds (TMPF) account and the SahulatCard acts as a debit card for this account. If the student then purchases a meal plan for the current term, purchases using the TMPF account take advantage of that meal plan's discount. But these purchases are subject to GST. If the TMPF account runs out of funds, then food purchases are automatically charged to the Flex Rupees account, which means that no discount applies.

On the next pages, there is a UML model, C++ class declaration for a composite object SahulatCard and class declarations for Date and Money ADTs that are used by SahulatCard objects and components.
Questions 1 through 5 ask you to implement code fragments of the composite object SahulatCard. Your answers to these questions should agree with one another (e.g., they should compile, link, and work together). For full marks, encapsulate data and avoid redundant code.

Question 1 – Composite Object Initialization [10 points]

Provide the C++ definition for the constructor of the SahulatCard class. The constructor should initialize all data members and should produce an object that adheres to the provided UML model. It should:
- have a unique barcode value
- have no PIN
- be inactive
- have an expiry date of 2 years from today
- have a Flex Account with a balance of PKR 0.00
- have no other accounts

The initialization of object members should be as efficient as possible.

Question 2 – Essential Methods [10 points]

a) Consider the following methods for the SahulatCard class: default constructor, destructor, copy constructor, assignment operator=. For each, state whether the compiler-generated version (if it exists) would be appropriate; explain your answer. (Credit is awarded only for correct explanations; not for correct determinations.)

b) Provide the C++ definition of the assignment operator= for the Meal Plan Funds class (regardless of whether it makes sense to implement this operator).

Question 3 – Abstract Base Class [10 points]

a) Provide the C++ declaration for the Account class.
- Your declaration should conform to the UML model
- Define Account to be an abstract base class.
- Add an accessor and a mutator (not shown in the UML model) that allow derived classes to access and set the value of data member balance. These methods should not change the public interface of Account (as modelled in UML). You **may** use these methods in your answers.
- Use const appropriately.

b) Provide a C++ default implementation for the method Account::purchase(). Assume no discounts and no GST. If there are sufficient funds in the account, the method subtracts the amount (given by parameter) from the account's balance and returns true. If there are insufficient funds in the account, then no funds are subtracted and the method returns false.

Question 4 – Derived Class [10 points]

a) Provide the C++ definition for the constructor for class Meal Plan Funds. (Think about whether a Meal Plan Funds object should refer to a shared Meal Plan object or should refer to its own copy of a Meal Plan object.)

b) Provide a C++ definition for the method MealPlanFunds::purchase() that applies the meal plan's discount to the purchase. Assume that a discount is a whole number (e.g., value 50 represents a 50% discount).

c) Provide a C++ definition for the method MealPlanFunds::endMealPlan(toAccount) that removes the Meal Plan and transfers any unused funds to the given account.

Question 5 – Polymorphic Code [10 points]

Provide the C++ definitions for the following SahulatCard methods.

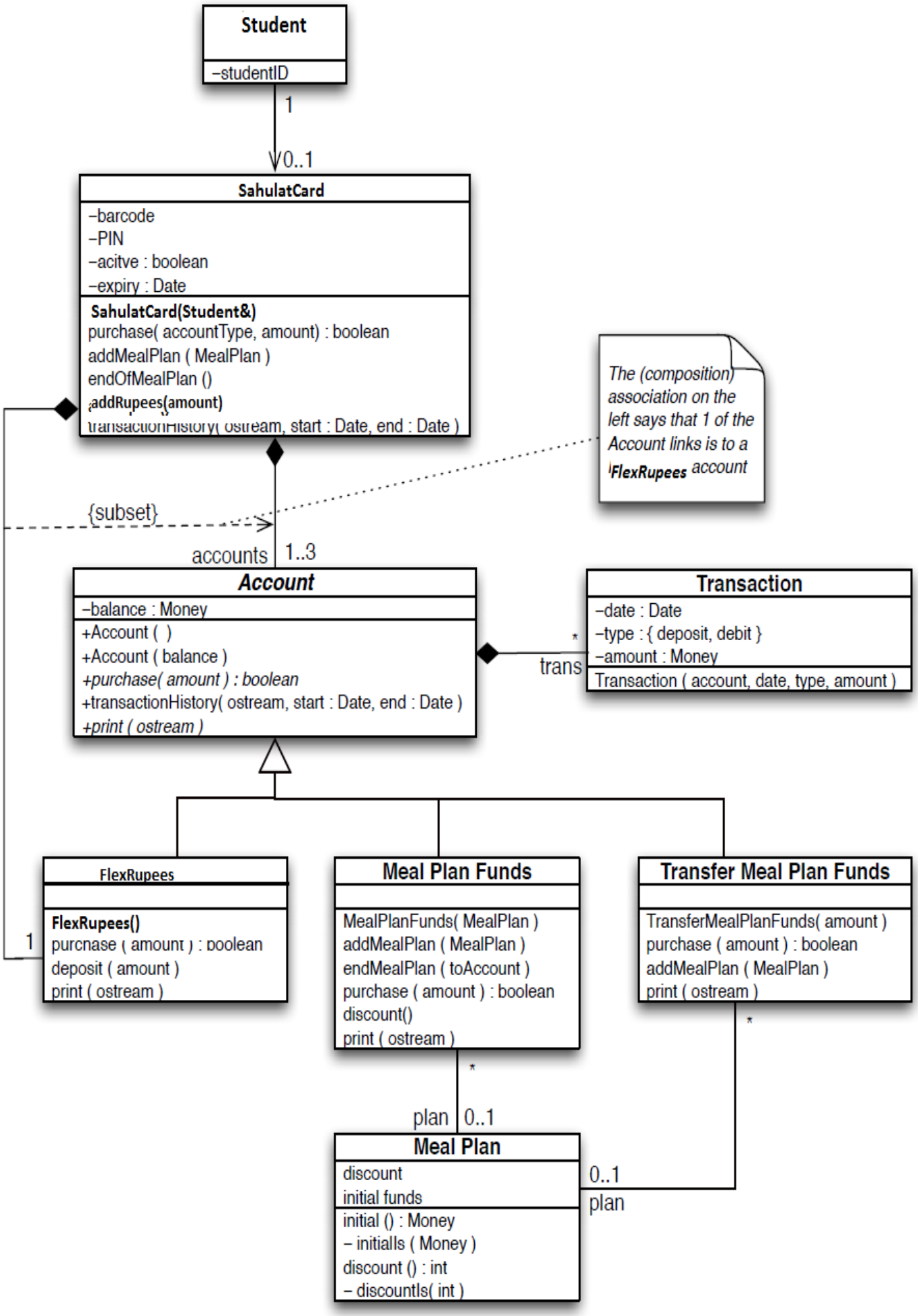
a) **`SahulatCard::purchase()`** - If the purchase is made to a Meal Plan Funds account or a Transfer Meal Plan Funds account and there are insufficient funds, then instead charge the purchase to the Flex Rupees account.

b) **`SahulatCard::addRupees()`** - Add funds to the Flex Rupees account.

c) **`SahulatCard::transactionHistory()`** - For each of the SahulatCard's accounts, stream all transactions that were conducted between the given start and end dates, inclusive.

SahulatCard UML model used in Questions 1-5

`On Page 3`


Student
-studentID
1
0..1
SahulatCard
-barcode
-PIN
-acitve : boolean
-expiry : Date
SahulatCard(Student&)
purchase(accountType, amount) : boolean
addMealPlan (MealPlan)
endOfMealPlan ()
addRupees(amount)
transactionHistory(ostream, start : Date, end : Date)
The (composition) association on the left says that 1 of the Account links is to a FlexRupees account
{subset}
accounts
1..3
Account
-balance : Money
+Account ( )
+Account (balance)
+purchase(amount) : boolean
+transactionHistory(ostream, start : Date, end : Date)
+print (ostream)
Transaction
-date : Date
-type : { deposit, debit }
-amount : Money
Transaction (account, date, type, amount)
*
trans
FlexRupees
FlexRupees()
purchase (amount) : boolean
deposit (amount)
print (ostream)
1
Meal Plan Funds
MealPlanFunds(MealPlan)
addMealPlan (MealPlan)
endMealPlan (toAccount)
purchase (amount) : boolean
discount()
print (ostream)
Transfer Meal Plan Funds
TransferMealPlanFunds(amount)
purchase (amount) : boolean
addMealPlan (MealPlan)
print (ostream)
*
*
plan
0..1
Meal Plan
discount
initial funds
initial () : Money
- initialIs (Money)
discount () : int
- discountIs(int)
0..1
plan

SahulatCard C++ class declaration used in Questions 1-5

```
#include <iostream>
#include "Account.h"
#include "Money.h"
#include "Date.h"
enum AccountType {FLEX, MEAL, TRANSFER};
class SahulatCard {
private:
static const int NUM = 3;
Student &s;
long barcode;
int* PIN;
bool active;
Date expiry;
Account* accounts[NUM];
/* Assume that AccountType (above) indices denote type of Account (i.e.,
accounts[FLEX] is a FlexRupees account */
long newBarcode();
public:
SahulatCard( const Student&); // constructor
bool purchase( AccountType, const Money &amount);
// debit amount from specified account; return success code
void addMealPlan( const MealPlan& ); // add meal plan to SahulatCard
void endOfMealPlan();
/*remove meal plan; transfer unused funds to Transfer Meal Plan    account */
void addRupees( const Money &amount );
// add money to FlexRupees account
void transactionHistory(ostream &out, const Date &start, const Date &end)const;
// stream transactions for all accounts between start and end dates
};
```

Date C++ class declaration; may be useful in Questions 1-5

```
#include <string>
#include <iostream>
class Date {
public:
Date(); // returns new Date = today
Date (int day, string month, int year); // constructor
int day() const; // accessor
string month() const; // accessor
int year() const; // accessor
Date incDays (long) const; // increments Date by some number of days
Date incMonths (int) const; // increments Date by some number of months
Date incYears (int) const; // increments Date by some number of years
};
bool operator== (const Date&, const Date&);
bool operator!= (const Date&, const Date&);
bool operator< (const Date&, const Date&);
bool operator<= (const Date&, const Date&);
bool operator> (const Date&, const Date&);
bool operator>= (const Date&, const Date&);
ostream& operator<< (ostream&, const Date&);
istream& operator>> (istream&, Date&);
```

Money C++ class declaration; may be useful in Questions 1-5

```
#include <string>
#include <iostream>
class Money {
public:
explicit Money( long Rupees = 0, long Paisas = 0);
// Constructor with specified money value
Money Rupees() const; // Accessor - Rupee value
Money Paisas() const; // Accessor - Paisa value
Money operator+ (const Money& n) const; // Add two money values; return result
Money operator- (const Money& n) const; // Subtract n; return result
Money operator* (int f) const; // Multiply by f%; return result
};
bool operator== (const Money& m, const Money &n);
bool operator!= (const Money& m, const Money &n);
bool operator< (const Money& m, const Money &n);
bool operator<= (const Money& m, const Money &n);
bool operator> (const Money& m, const Money &n);
bool operator>= (const Money& m, const Money &n);
istream& operator>> (istream&, Money&);
ostream& operator<< (ostream&, const Money&);
```

ASSIGNMENT SUBMISSION POLICY

- Assignment should be submitted INDIVIDUALLY.
- All submissions through SLATE. No emailing of assignments.
- All questions combined in one project file (including all **.cpp** and **.h** files).
- The project file is zipped into one folder.
- Zipped file should be: section followed by roll number e.g. B142033.zip
- Corrupt file or un-compliable submission will result in a zero.
- Internet/SLATE not working is not an acceptable excuse.
- No late day's policy.

Note!!!

- Taking other's solution and copied the solution is not acceptable.
- Academic integrity is very important to me and I regard ethical values more than anything in this world.
- I take plagiarism seriously. Ignorance is not an acceptable excuse. Punishment for this offence varies from negative marking to "F" in this course.

"**Honesty is the best policy**"