**#1 Chapter 11, Problem 4**

**use** strict**;**

**use** warnings**;**

# Frank Mock CS 122 Test 2

# Chapter 11, Problem 4

# This program provides a function that calculates and

# returns the monthly payment on a loan assuming a

# conventional loan where the interest is compounded

# monthly

# Get the loan amount from the user

**print**"What is the loan amount?"**;**

**chomp(my** $amount **=** <STDIN>**);**

# Get the interest rate from the user

**print**"What is the annual interest rate (from 1 to 100)?"**;**

**chomp(my** $rate **=** <STDIN>**);**

# Get the number of years of the loan from the user

**print**"Enter the number of years of the loan"**;**

**chomp(my** $years **=** <STDIN>**);**

# Define variables that hold the value of loan details

**my** $P **=** $amount**;** # principal, the initial amount of the loan

**my** $I **=** $rate**;** # The annual interest rate ( from 1 to 100 )

**my** $L **=** $years**;** # The length (in years) of the loan

# Hash that contains the loan info to pass to function

**my** %loanInfo **=** **(**"Principal" **=>** $P**,**

"Interest" **=>** $I**,**

"Length" **=>** $L**);**

# This function takes a hash that contains loan details as a

# parameter and returns the monthly payment amount

**sub** calcLoanPayment

**{**

**my** %h **=** @\_**;**

**my** $p **=** $h**{**'Principal'**};**

**my** $i **=** $h**{**'Interest'**};**

**my** $l **=** $h**{**'Length'**};**

**my** $J **=** $i**/(**12**\***100**);** # Monthly interest rate in decimal form

**my** $N **=** $L**\***12**;** # Number of months over which the loan is amoritzed

**my** $M **=** $p**\*(**$J**/(**1 **-** **(**1 **+** $J**)\*\*-**$N**));**

**return** $M**;**

**}**

# Call the function calcLoanPayment and assign return value to variable

**my** $monthly\_payment **=** calcLoanPayment**(**%loanInfo**);**

# Print the monthly payment returned from the function

**printf(**"The monthly payment for a 3o year loan of $P at a rate of $I percent is \$%7.2f"**,** $monthly\_payment**);**

**Program run and output are on the next page.**

**Program run and output** **for** **Chapter 11, Problem 4**

What is the loan amount?710000

What is the annual interest rate (from 1 to 100)?4

Enter the number of years of the loan30

The monthly payment for a 3o year loan of 710000 at a rate of 4 percent is $3389.65

**Next problem on the following page**

**#2 Chapter 12, Problem 5**

**use** Data**::**Dumper**;**

**use** strict**;**

**use** warnings**;**

# Define a complex data structure containing keys:

# Name, SSN, Friends[], and a Grades key that

# contains the keys Science, Math, English

**my** $structure **=** **{**

Name **=>** **undef,**

SSN **=>** **undef,**

Friends **=>** **[],**

Grades **=>**

**{**

Science **=>** **[],**

Math **=>** **[],**

English **=>** **[]**

**}**

**};**

# Assign values to the data stucture defined above

$structure**->{**'Name'**}=**"John Smith"**;**

$structure**->{**'SSN'**}=** "123-45-6789"**;**

$structure**->{**'Friends'**}=[**"Tom"**,** "Bert"**,** "Nick"**];**

$structure**->{**'Grades'**}->{**'Math'**}=[**102**,** 100**,** 99**];**

$structure**->{**'Grades'**}->{**'Science'**}=[**90**,** 100**,** 98**];**

$structure**->{**'Grades'**}->{**'English'**}=[**85**,** 98**,** 99**];**

# Print out the values contained in the data structure

**print**"Name is $structure->{'Name'}\n"**;**

**print**"Social Security number is $structure->{'SSN'}\n"**;**

**print**"Friends are "**;**

# Loop through the array of friends

**for(my** $j **=** 0**;** $j **<** 3**;** $j**++)**

**{**

**print**"$structure->{'Friends'}[$j] "**;**

**}**

**print**"\nGrades are:\n"**;**

**print**"\tScience: "**;**

# Loop through and print the Science grades

**for(my** $i **=** 0**;**$i **<** 3**;** $i**++)**

**{**

**print**"$structure->{'Grades'}->{'Science'}[$i] "**;**

**}**

# Loop through and print the Math grades

**print**"\n\tMath: "**;**

**for(my** $k **=** 0**;** $k **<** 3**;** $k**++)**

**{**

**print**"$structure->{'Grades'}->{'Math'}[$k] "**;**

**}**

# Loop through and print the Enlish grades

**print**"\n\tEnglish: "**;**

**for(my** $l **=** 0**;** $l **<** 3**;** $l**++)**

**{**

**print**"$structure->{'Grades'}->{'English'}[$l] "**;**

**}**

**Output for #2 Chapter 12, Problem 5:**

Name is John Smith

Social Security number is 123-45-6789

Friends are Tom Bert Nick

Grades are:

Science: 90 100 98

Math: 102 100 99

English: 85 98 99

**Next problem on the next page**

**#3 Chapter 13, Problems 2, 4**

**use** strict**;**

**use** warnings**;**

# Frank Mock CS 122 Test 2

# Chapter 13, Problems 2 and 4

# This program simulates ATM machine transactions

**package** Checking**;**

**my** $balance **=** 0**;** # Lexical private variable

# Returns the value of the private balance variable

**sub** get\_balance **{ return** $balance**;** **}**

# Deposit increases the value of $balance

**sub** deposit **{**

**my** $amount **=** **shift;**

$balance **+=** $amount**;**

**}**

# withdraw reduces the value of $balance

**sub** withdraw **{**

**my** $amt **=** **shift;**

$balance **-=** $amt**;**

**}**

# Write the current balance to file register.txt

**sub** saveBalance

**{**

**my** $date **=** **localtime();** # get the current date and time

# open register file for writing

**open(my** $fh**,** ">"**,** "register.csv"**)** **||** **die** "Could not save balance to file."**;**

**print** $fh "$balance,$date"**;**

**close** $fh**;**

**}**

# Get balance from the register.csv file

**sub** readBalanceRegister

**{**

**open(my** $fh**,** "<"**,** "register.csv"**)** **||** **die**"Could not read balance from file."**;**

#read each line of the register file into the @bal array

**chomp(my** @bal **=** <$fh>**);**

# First element is current balance from file

**my** $line **=** $bal**[**0**];**

# The balance and date are separated by a comma

**my** @tokens **=** **split(**/,/**,** $line**);**

# The first token is the balance from the file

$balance **=** $tokens**[**0**];}**

**package** main**;**

# Read the last balance saved to file

Checking**::**readBalanceRegister**();**

# menu sub routine displays a list of ATM choices

**sub** menu**{**

**print** <<EOF**;**

1. Deposit

2. Withdraw

3. Current Balance

4. Exit

EOF

**}**

**my** $amt **=** 0**;**

**my** $choice **=** 3**;** # Default choice is Current Balance

# Prompt the user to make a selection until choice 4

# is selected.

**while(**$choice **!=** 4**)**

**{**

menu**;**

**print**"Please select (1 - 4) from the menu: "**;**

**chomp(**$choice **=** <STDIN>**);** # Save users input to variable $choice

# Take action depending on the users choice

**if(**$choice **==** 1**)**

**{**

**print**"Enter deposit amount: "**;**

**chomp(**$amt **=** <STDIN>**);** # Save users input to variable $amt

Checking**::**deposit**(**$amt**);**

**}**

**elsif(**$choice **==** 2**)**

**{**

**print**"Enter amount to withdraw:"**;**

**chomp(**$amt **=**<STDIN>**);** # Save users input to variable $amt

Checking**::**withdraw**(**$amt**);**

**}**

**elsif(**$choice **==** 3**)**

**{**

$amt **=** Checking**::**get\_balance**();**

**print**"The current balance is \$$amt\n"**;**

**}**

**else** # End of program

**{**

# Save balance to file before exiting

Checking**::**saveBalance**();**

**print**"Goodbye"**;**

**}**

**}**

**Program runs and ouput produced**

1. Deposit

2. Withdraw

3. Current Balance

4. Exit

Please select (1 - 4) from the menu: 3

The current balance is $150

1. Deposit

2. Withdraw

3. Current Balance

4. Exit

Please select (1 - 4) from the menu: 1

Enter deposit amount: 200

1. Deposit

2. Withdraw

3. Current Balance

4. Exit

Please select (1 - 4) from the menu: 3

The current balance is $350

1. Deposit

2. Withdraw

3. Current Balance

4. Exit

Please select (1 - 4) from the menu: 2

Enter amount to withdraw:50

1. Deposit

2. Withdraw

3. Current Balance

4. Exit

Please select (1 - 4) from the menu: 3

The current balance is $300

1. Deposit

2. Withdraw

3. Current Balance

4. Exit

Please select (1 - 4) from the menu: 4

Goodbye

**Then after running the program again it reads the balance that was saved to file:**

1. Deposit

2. Withdraw

3. Current Balance

4. Exit

Please select (1 - 4) from the menu: 3

The current balance is $300

1. Deposit

2. Withdraw

3. Current Balance

4. Exit

Please select (1 - 4) from the menu: 4

Goodbye

**#3 Chapter 13, Problem 6**

A.)

The user has direct access to the symbols from the Checking module since they were declared using the **our** keyward and not the **my** keyword. By using the our keyword the array, @EXPORT\_OK and hash, %EXPORT\_TAGS are publically excessible to any script that uses the Checking modue. The following is how you would import each data structure from package main.

**use** Checking; **# To use Checking module**

**my** @copy **=** @Checking::EXPORT\_OK**;** #copy array from Checking module

**my** %hashCopy **=** @Checking::EXPORT\_TAG**;** #copy hash from Checking module

It’s not necessary to copy each into a separate variable as in my examples above, instead you could use each directly.

**B.)**

No, Each module should be defined in their own file with the .pm extention. This ensures that the modules are modular and extensible.

**Go to next page for next problem**

**#4 Chapter 14, Part 2 problems 1, 2**

# Student.pm

# Chapter 14 Part 2

# by Frank Mock, CS 122, Test 2

# This Perl file is the definition of a Student class

**package** Student**;** # The name of the class

**use** strict**;**

**use** warnings**;**

**our** $VERSION **=** "0.1"**;**

# The constructor method is named new by convention

**sub** new

**{**

# The name of the class is given to $class when object is created

**my** $class **=** **shift;**

# $self is an anonymous hash used for instance data (dynamic data)

**my** $self **=** **{**

"Courses" **=>** **[]**

**};**

# use the bless function to turn $self hash in to an object

**bless(** $self**,** $class **);**

# return a reference to the blessed class

**return** $self**;**

**}**

# setter and getter for name

**sub** name

**{**

**my** **(**$self**,** $name**)** **=** @\_**;**

# if a value for $name is passed in, assign it to $name

$self**->{**name**}** **=** $name **if** **defined** $name**;**

**return** $self**->{**name**};**

**}**

# setter and getter for major

**sub** major

**{**

**my** **(**$self**,** $major**)** **=** @\_**;**

# if a value for $major is passed in, assign it to $major

$self**->{**major**}** **=** $major **if** **defined** $major**;**

**return** $self**->{**major**};**

**}**

# setter for courses

**sub** setCourses

**{**

**my** **(**$self**,** @courses**)** **=** @\_**;**

**my** $size **=** $#courses **+** 1**;**

**for(my** $g **=** 0**;** $g **<** $size**;** $g**++)**

**{**

$self**->{**"Courses"**}->[**$g**]** **=** $courses**[**$g**];**

**}**

**return** 1**;**

**}**

# setter and getter for address

**sub** address

**{**

**my** **(**$self**,** $address**)** **=** @\_**;**

# if a value for $address is passed in, assign it to $address

$self**->{**address**}** **=** $address **if** **defined** $address**;**

**return** $self**->{**address**};**

**}**

# setter and getter for id

**sub** id

**{**

**my** **(**$self**,** $id**)** **=** @\_**;**

# if a value for $id is passed in, assign it to $id

$self**->{**id**}** **=** $id **if** **defined** $id**;**

**return** $self**->{**id**};**

**}**

# setter and getter for start\_date

**sub** start\_date

**{**

**my** **(**$self**,** $start\_date**)** **=** @\_**;**

# if a value for $start\_date is passed in, assign it to $start\_date

$self**->{**start\_date**}** **=** $start\_date **if** **defined** $start\_date**;**

**return** $self**->{**start\_date**};**

**}**

# This method prints all the value of all instance variables

**sub** show\_student

**{**

**my** **(**$self**)** **=** @\_**;**

**print**"Name: $self->{name}\n"**;**

**print**"ID: $self->{id}\n"**;**

**print**"Major: $self->{major}\n"**;**

**my** $s **=** $self**->{**"Courses"**};**

**print**"Courses:"**;**

**for(my** $i **=** 0**;** $i **<** 4**;** $i**++)**

**{**

**print(**" "**,**$self**->{**"Courses"**}->[**$i**]);**

**}**

**print**"\nAddress: $self->{address}\n"**;**

**print**"Start Date: $self->{start\_date}"**;**

**return** 1**;**

**}**

1**;** #module must return 1

**And here is the file that contains the main method that uses the Student class to create three student modules:**

**use** Student**;**

**use** Data**::**Dumper**;**

**use** strict**;**

**use** warnings**;**

# testStudent.pl

# This program creates three Student objects and sets their instance

# variables. The show\_student method is called for each student

# which prints the values of each students instance variables.

main**(**@ARGV**);**

**sub** main

**{**

#Create three Student objects

# Student object 1 details

**my** @list **=** **(**"English"**,** "Science"**,** "Calculus"**,** "History"**);**

**my** $student **=** Student**->**new**();** #create an instance of a Student

$student**->**setCourses**(**@list**);**

$student**->**name**(**"Frank Mock"**);**

$student**->**major**(**"Computer Science"**);**

$student**->**address**(**"123 C Street Vacaville, CA 94536"**);**

$student**->**id**(**"101"**);**

$student**->**start\_date**(**"08/15/2012"**);**

$student**->**show\_student**();** # Prints the Student object to output

**print**"\n\n"**;** #print a new line to separate object print out

# Student object 2 details

**my** @list2 **=** **(**"German"**,** "Science"**,** "Algebra"**,** "History"**);**

**my** $student2 **=** Student**->**new**();**

$student2**->**setCourses**(**@list2**);**

$student2**->**name**(**"Henry Ford"**);**

$student2**->**major**(**"Computer Science"**);**

$student2**->**address**(**"456 S Street Dixon, CA 94586"**);**

$student2**->**id**(**"150"**);**

$student2**->**start\_date**(**"08/15/2013"**);**

$student2**->**show\_student**();**

**print**"\n\n"**;** #print a new line to separate object print out

# Student object 3 details

**my** @list3 **=** **(**"French"**,** "Biology"**,** "CalculusII"**,** "Cooking"**);**

**my** $student3 **=** Student**->**new**();**

$student3**->**setCourses**(**@list3**);**

$student3**->**name**(**"Alice Cooper"**);**

$student3**->**major**(**"Computer Science"**);**

$student3**->**address**(**"789 Sun Street Sunnyville, CA 94580"**);**

$student3**->**id**(**"160"**);**

$student3**->**start\_date**(**"08/15/2013"**);**

$student3**->**show\_student**();**

**}**

**Here is the program run and output:**

Name: Frank Mock

ID: 101

Major: Computer Science

Courses: English Science Calculus History

Address: 123 C Street Vacaville, CA 94536

Start Date: 08/15/2012

Name: Henry Ford

ID: 150

Major: Computer Science

Courses: German Science Algebra History

Address: 456 S Street Dixon, CA 94586

Start Date: 08/15/2013

Name: Alice Cooper

ID: 160

Major: Computer Science

Courses: French Biology CalculusII Cooking

Address: 789 Sun Street Sunnyville, CA 94580

Start Date: 08/15/2013