Ruby
Practical 9
Student Name- Vipin Hans
Student Number- 17201230

1) Use ActiveRecord to set up a database with a number of records in it. The database should contain records for library users (with fields for name, age, books borrowed) and library books (with fields for borrower, title, borrowedwhen, dueback). Note, it should be set up so that a borrower can borrow many books. Create classes for these objects and then use ***find*** to print out the atributes of the various records that you have created.
Answer-
I've created **createdatabase.rb** program which creates database and tables for library users and books. This has been setup that borrower can borrow many books.

```
class LibraryBook < ActiveRecord::Base
#	set_primary_key "borrower_id"
	belongs_to :library_users
end

class LibraryUser < ActiveRecord::Base
#	add_foreign_key :library_users, :library_books, column: :user_id, primary_key:
:borrower_id
	has_many :library_books
end

if (LibraryUser.table_exists? || LibraryBook.table_exists?)
	puts "table exists"
	else ActiveRecord::Schema.define do
			create_table :library_users do |table|
			table.column :user_id, :integer
			table.column :name, :string
			table.column :age, :integer
		end

		create_table :library_books do |table|
			table.column :library_user_id, :integer
			table.column :borrower, :string
			table.column :title, :string
			table.column :borrowedwhen, :string
			table.column :dueback, :string
		end
	end
end
```

The statements written below creates a user instance with a single user borrowing multiple books this displaying one to many design.

```
user = LibraryUser.create(:user_id => 1,:name => 'Mike',:age => 27)
```

```
user.library_books.create(:borrower => 'Mike', :title => 'Harry Potter', :borrowedwhen
=> '16 Nov 2017', :dueback => '20 Nov 2017')

user.library_books.create(:borrower => 'Mark', :title => 'Harry Potter 2',
:borrowedwhen => '18 Nov 2017', :dueback => '22 Nov 2017')
user = LibraryUser.create(:user_id => 2,:name => 'Mikessss',:age => 21)

user.library_books.create(:borrower => 'Max', :title => 'Making the game',
:borrowedwhen => '10 Nov 2017', :dueback => '18 Nov 2017')

user.library_books.create(:borrower => 'Justin', :title => 'Orphan', :borrowedwhen =>
'20 Nov 2017', :dueback => '28 Nov 2017')
```

I've also used various format of ***find*** functions to evaluate and find the table entries using system generated id numbers and string such as name and book title.

```
p LibraryUser.all.each { |u| puts 'User name '+ u.name + 'borrowed books age is ' +
u.age.to_s }
p LibraryBook.all
puts "Diaplying books using sql where clause"
p LibraryBook.where("borrower = 'Max' AND title = 'Harry Potter'")
puts "Diaplying records using find function"
p LibraryBook.find(1)
puts "Diaplying records using find by function"
p LibraryBook.find_by_title("Harry Potter 2")
p LibraryBook.find_by_title("Making the game")
p LibraryBook.find_by_borrowedwhen("20 Nov 2017")
```

2) Do something really smart with the blocks and lambda stuff shown in the notes.
Answer-
I've created another program **lambda.rb** which makes use of lambda function in 2 scenarios. First one just simply takes a integer value as a parameter and evaluates to square of a number and second one takes a regular expression and finds if its included in the predefined string.

```
calculatesquareofnumber=lambda{|x| puts x*x}
calculatesquareofnumber.call(100)

a=/Hell/
str="Hello world"
comparestring=lambda{|a,str|
if((str=~a) !=nil)
        puts "The regular expression matches the string.."
elsif
        puts "Its not the part of string"
end
}
comparestring.call(a,str)
```