1. (10%) A computer has a bus with a 25 nsec cycle time, during which it can read or write a 32-bit word from memory. The computer has a disk that uses the bus and runs at 40 Mbytes/sec. The CPU normally fetches and executes one 32-bit instruction every 25 nsec. How much does the disk slow down the CPU?

2. (10%) Which of the following services use TCP?

(i) DHCP
(ii) SMTP
(iii) HTTP
(iv) TFTP
(v) FTP

Choose one of the answers below:

(A) (i) and (ii)
(B) (i), (iii), and (v)
(C) (ii), (iii), and (v)
(D) (i), (ii), and (iv)

3. (10%) Consider the relation Movie with the following attributes:
(title, year, length, filmType, studioName, studioCountry, starName)

Show functional dependencies and how you progress from 1NF through 2NF to a set of 3NF relations. At each stage show the primary key and any foreign keys of each relation and state assumptions that you make about any of the relationships between the columns of data.

4. (15%) Please write pseudo code for the function described as follows. There are 300 students, and 10 elective courses with 30 seats allowed for each elective course. The task is to allocate students to their preferred elective courses. Every student assigns the preference of the 10 courses from 1 to 10. Then, students would be randomly assigned a number from 1 to 300, with student "1" allocated her top choice. Student "2" would then get her top choice, and so on. If the top choice for a student, say "80", is already full when it gets to her round, she will then get her second choice (if that is full too, her third choice, and so on until she gets one course for that round). After the first round, each student would have got one course. Then the same process goes into the second round – this time starting from the last student, i.e. student "300". The same process goes on until all the courses are allocated. Please state any assumptions you think necessary in writing your code.

5. (10%) The following code is written as a dynamically typed, object-oriented language. It defines a recursive function that takes a nested array as its 'arr' parameter.

```
def fun_print(arr) {
  if (length(arr) > 1) {
    fun_print(arr[1..length(arr)-1])
  }

  if arr[0].type_of(String) {
    print(arr[0] + ' ')
  elsif arr[0].type_of(Array)
    fun_print(arr[0])
  }
}
```

What would you expect this function to output if called as follows:

```
fun_print( [['A', 'B', 'C']], 'D', ['E', 'F']] )
```

(A) A D E B F C　　(C) A B C D E F　　(E) F E D C B A

(B) E F D A B C　　(D) A B C E F D　　(F) D E F A B C

To answer the next three questions, pick from the following data structures:

1. Hash Table
2. Array
3. Stack
4. Linked List
5. Binary Tree
6. Queue

6. (10%) Which of the above data structures do we usually expect to be *indexed*? (i.e., you could directly retrieve any element by supplying an index value):

(A) 2, 3, and 5
(B) 4 and 6
(C) 1, 2, and 6
(D) 3, 4, and 5
(E) 2, 3, and 5
(F) 1 and 2

7. (10%) Which of these structures would we use to keep a list of recently used files, such that we can quickly retrieve and remove files in *most recently used order*?

(A) 1
(B) 2
(C) 3
(D) 4
(E) 5
(F) 6

8. (10%) Which of these data structures would we use to enter a list of tasks, such that we can retrieve and remove the tasks in the *same order they were entered*?

(A) 1
(B) 2
(C) 3
(D) 4
(E) 5
(F) 6

9. (15%) The following code is written in a dynamic object-oriented programming (OOP) language. Refactor and improve the code without changing its function! (20%)

```
class Student
  var name
  var must_teach

  def initialize(full_name)
    name = full_name
    must_teach = false
  end

  def notify_enroll(semester)
    puts("{{name}} must enroll for classes in {{semester}}")
  end
end

class Lecturer
  var name
  var must_teach
  var must_research

  def initialize(full_name)
    name = full_name
    must_teach = true
    must_research = false
  end

  def notify_grades(semester)
    puts("{{name}} must submit grades for {{semester}}")
  end
end

class Professor
  var name
  var must_teach
  var must_research

  def initialize(full_name)
    name = full_name
    must_teach = true
    must_research = true
  end

  def notify_grades(semester)
    puts("{{name}} must submit grades for {{semester}}")
  end

  def notify_grant(semester)
    puts("{{name}} must submit a grant in {{semester}}")
  end
end
```

You must rewrite the above code as follows:

- Use object-orientated principles to remove repetition in the code
- Your new code must continue working with existing test code (see top of next page)
- Follow the OOP syntax of the programming language (see bottom of next page)

Here is working test code that must still work after your refactoring:

```
# TEST CODE
lee = Student.new('Li-Hsieh Huang')
hsu = Lecturer.new('Powei Hsu')
lu = Professor.new('Peishan Lu')

everyone = [lee, hsu, lu]

for person in everyone
  if (not person.must_teach)
    person.notify_enroll('Spring 2016')
  end

  if (person.must_teach)
    person.notify_grades('Fall 2015')
  end

  if (person.must_teach) && (person.must_research)
    person.notify_grant('Spring 2016')
  end
end

# TEST OUTPUT
# => Li-Hsieh Huang must enroll for classes in Spring 2015
# => Powei Hsu must submit grades for Fall 2014
# => Peishan Lu must submit grades for Fall 2014
# => Peishan Lu must submit a grant in Spring 2015
```

Here is sample OOP code to help you understand the syntax of inheritance:

```
# SAMPLE OOP CODE:
class Automobile
  var brand                        # var defines a public instance variable
  var mode

  def initialize(brand_name) # initialize is a constructor method
    brand = brand_name
    mode = 'ground'
  end
end

class Car < Automobile         # Car inherits from Automobile class
  var model
  var num_wheels

  def initialize(brand_name)
    super(brand_name)          # Calls the parent's constructor method
    num_wheels = 4
  end

  def describe()
    puts "The {{brand}} {{model}} has {{num_wheels}} wheels"
  end
end

my_car = Car.new('Luxgen')
my_car.model = 'U6 Turbo'
my_car.describe

# OUTPUT
# => "The Luxgen U6 Turbo has 4 wheels"
```