NAME : Answers

Open book, open notes, but please do not ask questions.
Write all answers on these sheets.

question	1	2	3	4	5	6	total
points							
maximum	15	15	20	20	15	15	100

1. To process preferences we create forms, where preferences are listed using checkboxes. Write code for the function

```
def WriteForm(n,L,p):
   """
   Returns the string that defines an html form.
   The input parameters are
     n is the name of the checkbox,
     L is a list of preferences,
     p contains the name of the string which
       will handle the input form.
   The value of each element in the checkbox
   is its position in the list.
   One submit button concludes the form.
   """
```

Answer:

```
   s = "<form method=\"post\" action=\"%s\">" % p
   for i in range(0,len(L)):
      c = "\n<p><input name = \"%s\" type = checkbox value = %s>" \
          % (n,str(i))
      c = c + str(L[i])
      s = s + c
   s = s + "\n<p><input type=\"submit\" value = \"submit\">"
   s = s + "\n</form>"
   return s
```

/15

2. To order a certain good in a certain quantity, an html form has two input elements: `good` for the name of the good, and `quantity` for the number of items ordered.

   The script which handles this form writes plaintext to the web page. If the name of the good is missing it writes `"please provide name of good"`. The default value of `quantity` is one. If the name of the good is provided, the script confirms writing `"ordered <n> of <good>"`, where `n` is the quantity and `good` the name of the good.

   Write the code of the script which handles this form in the way described above.

   Answer:

```
#!/Library/Frameworks/Python.framework/Versions/Current/bin/python
import cgi
print "Content-Type: text/plain\n"
form = cgi.FieldStorage()
if not form.has_key('good'):
   print 'please provide name of good'
else:
   g = form['good'].value
   n = int(form['quantity'].value)
   print 'ordered %d of %s' % (n,g)
```

/15

3. Suppose that data to be entered in a database of used cars is stored on file. For each car there is one line on the file. For example, the file `cars.txt` could look like.

```
{'make':'Toyota', 'year':2001, 'price':6995.21 }
{'make':'Honda', 'year':2005, 'price':13798.25 }
```

   For every car we store its make, year and sales price in a dictionary with keys `'make'`, `'year'`, and `'price'`. Then running the script on the file `cars.txt` prints

```
(0, 'Toyota', 2001, 6995.21)
(1, 'Honda', 2005, 13798.25)
```

   Give the code to open the file `cars.txt` and to write the tuples to screen.

   Answer:

```
file = open('cars.txt','r')
key = 0
while True:
   L = file.readline()
   if L == '': break
   d = eval(L)
   tp = (key,d['make'],d['year'],d['price'])
   print tp
   key = key + 1
file.close()
```

/20

4. Consider a server to perform simple calculations. The server listens for one client to send an arithmetic expression, e.g.: `4*8`. The server computes the value (in this example `32`) and sends the result to the client.

   (a) For the client/server program to work, the server and the client have to agree on certain issues. What are these issues? Describe those elements *common* in the code for both client *and* server.

   Answer:

   The three issues client and server both have to agree on are (1) the IP address of the server, (2) the port number, and (3) the buffer size.

   The code for the server has the following definitions:

```
hostname = ''   # blank for any address
number = 11267  # number for the port
buffer = 80     # size of the buffer
```

   The code for the client has the following definitions:

```
hostname = 'localhost'  # on same host
number = 11267          # same port number
buffer = 80             # size of the buffer
```

   (b) Write code for the function below:

```
def compute(server,buffer):
   """
   On input are in server the server socket
   and in buffer the buffer size.
   The server accepts connection from a client
   and waits for an arithmetic expression.
   Then the server sends the result of the
   expression to the client.
   """
```

   Answer:

```
   client, client_address = server.accept()
   e = client.recv(buffer)
   print 'received \"' + e + '\"'
   r = eval(e)
   client.send(str(r))
```

/20

5. Suppose we want to develop a screensaver. We view the screen as a grid of rectangles. Threads determine grayscales of each rectangle at random, after sleeping between 1 and 6 time units before generating a new random grayscale. To develop this screensaver, we first simulate the multithreaded generation of the grayscales.

   The main program prompts for the number of rows and columns. The names of the threads are tuples (i, j) with i and j the row and column of the rectangular area of the screen for which the thread will generate grayscales.

   Using `ScreenThread` as name of the class which defines the behavior of the threads, write code for the main program.

   Answer:

```
def main():
   n = input('give #rows : ')
   m = input('give #columns : ')
   T = []
   for i in range(0,n):
      for j in range(0,m):
         name = str((i,j))
         T.append(ScreenThread(name))
   print 'starting the threads'
   for t in T: t.start()
   print 'threads have started'
```

/15

6. We can measure the complexity of a web page by counting the < (less than) symbols. Write code for the following function:

```
def CountLT(url):
   """
   Opens the web page defined by the url and returns the number
   of '<' symbols found on the web page.
   Returns -1 if opening of the web page fails.
   """
```

   Answer:

```
   import urllib
   try:
      cnt = 0
      f = urllib.urlopen(url)
      while True:
         d = f.read(1)
         if d == '': break
         if d == '<': cnt = cnt + 1
      return cnt
   except:
      return -1
```

/15