Question 1. [3 MARKS]

For each block of code in this question, write its output in the box below it. If it would generate an error, say so, and give the reason for the error.

Part (a) [2 MARKS]

```
inner = [1, 2, 3]
nested = [inner, inner]
print nested[0]
```

```
[1, 2, 3]
```

```
# continued from above
nested[0].append(4)
print nested
```

```
[[1, 2, 3, 4], [1, 2, 3, 4]]
```

Part (b) [1 MARK]

```
inner2 = [4, 5, 6]
nested2 = [inner2[:], inner2[:]]
nested2[0].insert(3, 7)
print nested2
```

```
[[4, 5, 6, 7], [4, 5, 6]]
```

Question 2. [3 MARKS]

In A2, a `board` was a `list of lists of strs` where each inner list represented a row on the board. Complete the following function, according to its docstring description.

```
def display_column(board, col):
    '''(list of lists of strs, int) -> NoneType
    Print column col of the board.  You may assume: 0 <= col < len(board)'''

    for row in range(len(board)):
        print board[row][col]
```

Question 3. [5 MARKS]

Complete the following function according to its docstring description.

```
def num_start_with(sentences, letter):
    '''(list of lists of strs, str) -> int
    The strs in the lists in sentences are lowercase words
    (e.g., [['hi', 'there'], ['this', 'is', 'fun'], ['hooray']])
    and letter is a single lowercase letter.
    Return the number of words from sentences that start with letter.'''

    count = 0

    for word_list in sentences:
        for word in word_list:
            if word.startswith(letter):
                count != 1
    return count
```

You may use the space below for rough work. This section will not be marked unless you clearly indicate the part of your work that you want us to mark.

Question 4. [13 MARKS]

An employee is keeping track of the dates he is scheduled to work in a file, where each line is of the form:
`YEAR,MONTH,DAY`
`YEAR`s are 4-digits integers; `MONTH`s and `DAY`s are each 2-digit integers.

Here is a sample "schedule file":

```
2012,03,15
2012,03,18
2012,04,01
2012,10,10
```

A "schedule list" is generated based on a "schedule file" like the one describe above.

The "schedule list" for the file above is:
`["15-03-2012", "18-03-2012", "01-04-2012", "10-10-2012"]`.
Note: the date format in the file differs from the list (e.g., `2012,03,15` as compared to `15-03-2012`).

Part (a) [6 MARKS] Complete the following function according to its docstring description.

```
def get_schedule_list(schedule_file):
    '''(file) -> list of strs
    Return a "schedule list" for the "schedule file" schedule_file.'''

    dates = []

    for line in f:
        date_list = line.strip().split(',')
        date = "%s-%s-%s" % (date_list[2], date_list[1], date_list[0])
        dates.append(date)

    return dates
```

Part (b) [4 MARKS] Complete the following function according to its docstring description.

```
def is_booked(schedule_list, proposed_date):
    '''(list of strs, str) -> bool
    proposed_date is a date in the same format as those in a "schedule list".
    Return True if proposed_date is booked (employee is scheduled to work)
    according to "schedule list" schedule_list, and return False otherwise.'''

    found = False
    for date in schedule_list:
        if date == proposed_date:
            found = True
    return found
```

Part (c) [3 MARKS] In the question below, fill in the box with python code that will make the program behaviour match the comments. You may **not** make any other changes to the code.

```
if __name__ == '__main__':
    schedule_file = open('schedule.txt')
    schedule_list = get_schedule_list(schedule_file)

    # Continually prompt the user (using raw_input) to enter a date (in DD-MM-YYYY format)
    # until they enter a date that is not booked.

    date = raw_input("Enter a date: ")}
    while is_booked(schedule, date):
        date = raw_input("Enter a date: ")

    print "The date %s is not booked.", % (date)
```