Introduction to Programming (Python) (X_401096)
Practice Exam

The total number of points on this exam is 44. Your grade G follows from your point total P by $G = P \,/\, 44 \times 9 + 1$.

Problem 1:
Given are the following classes `A` and `B`:

```
1  class A:
2      def __init__(self, an_int):
3          self.g = an_int
4
5      def increment(self):
6          self.g += 1
7
8          if self.g >= 10:
9              self.g = 0
10
```

```
1  class B:
2      def __init__(self, a_obj):
3          self.a = a_obj
4
5      def increment(self):
6          self.a.increment()
7
8      def set(self, an_int):
9          self.a.g = an_int
10
```

Also given is the following program:

```
1  a = A(0)
2  b = B(a)
3
4  def println(a_obj, b_obj):
5      print("%d - %d" % (a_obj.g, b_obj.a.g))
6
7  println(a, b)
8  a.increment()
9  println(b.a, b)
10 a = A(9)
11 b.increment()
12 println(a, b)
13 b.a.g = a.g
14 a.increment()
15 println(a, b)
16 b.set(12)
17 a.increment()
18 println(a, b)
19 a = b.a
20 b.increment()
21 println(a, b)
```

(a) (5 points) What will be printed when this program is executed?

Continued on the next page →

A number is a perfect number if it is equal to the sum of its proper divisors (a divisor is a proper divisor if it is not equal to the number itself). For example, 28 is a perfect number, because its proper divisors are 1, 2, 4, 7 and 14, and $28 = 1 + 2 + 4 + 7 + 14$.

(b) (5 points) Write a program that checks for each positive integer < 5000 if that number is a perfect number, and prints all those numbers that are perfect numbers.

(c) (5 points) Write a function `def max_column_values(m)`. The parameter `m` is a 2-dimensional list of integers. The function should return the maximum value of each of the columns in the matrix `m`. Write this function without using the built-in function `max()`.

For example, if `m` is $\begin{bmatrix} 7 & 3 & 4 \\ 2 & 8 & 6 \\ 6 & 3 & 6 \end{bmatrix}$, the function should return `[7, 8, 6]`.

Now consider the following program:

```
 1  a = 4
 2  b = 3
 3
 4  def show(x, y):
 5      print("%d %d" % (x, y))
 6
 7  def m1(b):
 8      global a
 9      b -= 1
10      a = 2
11      show(a, b)
12      return b + a
13
14  def m2(a, c):
15      global b
16      b = c
17      a *= 2
18      c = m1(a)
19      show(b, c)
20      return b + c
21
22  show(b, a)
23  b = m1(b)
24  show(a, b)
25  a = m2(b, a)
26  show(b, a)
```

(d) (5 points) What will be printed when this program is executed?

Continued on the next page →

Problem 2:
Program subproblems in a separate method in the appropriate class.

Given is the following class `FootballPlayer`:

```
class FootballPlayer:
    def __init__(self, a_str, an_int1, an_int2, a_bool1, a_bool2):
        self.position = a_str           # the position of this player
        self.number_of_goals = an_int1  # number of goals scored
        self.number_of_games = an_int2  # number of games played
        self.injured = a_bool1          # whether this played is injured
        self.red_card = a_bool2         # whether this player has a red card
```

When applicable, you are allowed to solve subproblems by adding methods to this class.

(a) (2 points) Write a class `FootballTeam`, that can store any number of football players. The class should have an initializer method that initializes an empty football team. The class should also have a function called `add()`, that can add a football player to the team. You can add parameters to the method.

(b) (5 points) Add a method `def select_top_scorers(self)` to the class `FootballTeam`, that returns all players that are top scorers, in a new `FootballTeam` object. A player is a top scorer in case their average number of goals per game is at least 0.4.

(c) (5 points) Add a method `def remove_inactive_players(self)` to the class `FootballTeam`, that removes all inactive players from the team. A player is inactive in case they are injured or have a red card. In order to delete an element with index `i` from a list `l`, you can use `del l[i]`.

Given is that there exists a function `def sort_players(player)`. The parameter `players` should be a list containing only `FootballPlayer` objects. The function return s a new list that contains the same players as the original list, but sorted based on the average number of goals per game (from high to low).

(d) (6 points) Write a function `def select_formation(football_team)`. The parameter `football_team` is an object of type `FootballTeam`. The function should return a list that contains 11 football players. Those 11 players should be selected as follows:

- 1 player with the position `"keeper"`
- 4 players with the position `"defender"`
- 4 players with the position `"midfielder"`
- 2 players with the position `"striker"`

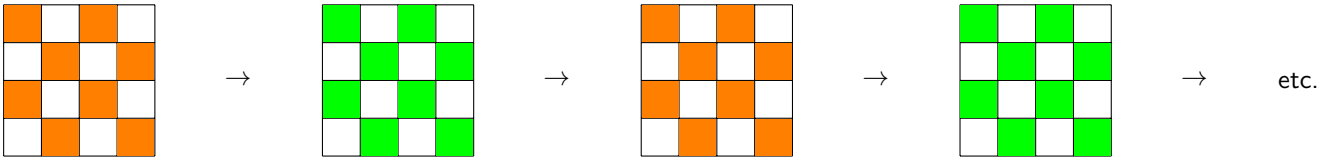
For each of these positions, you should select the players that have the highest average number of goals per game. You can assume that the football team contains enough players to make such a selection.

Continued on the next page →

Problem 3:
Given is the following program:

```
1  from snakelib import SnakeUserInterface
2
3  WIDTH = 10
4  HEIGHT = 10
5
6  # additional global variables go here
7
8  ui = SnakeUserInterface(WIDTH, HEIGHT)
9  ui.set_animation_speed(1)
10
11 def update():
12     # your code goes here
13
14 while True:
15     event = ui.get_event()
16     if event.name == "alarm":
17         update()
18         ui.show()
```

(6 points) Implement the function `def update()` in such a way that the program will show a checkerboard pattern continuously switching between wall pieces and snake pieces, as shown in the images below. You are also allowed to add global variables to the program.



In order to fill in a square at some coordinate (x,y) you can use `ui.place(x, y, ui.WALL)` or `ui.place(x, y, ui.SNAKE)`.

The width and height of the screen are determined by the constants `WIDTH` and `HEIGHT`. These constants will both have an int value ≥ 1. The program should work for any values that these constants can have.

End ▪