UNIVERSITEIT VAN AMSTERDAM 2016-2017

minor programmeren

Tentamen
Programmeren 2

vrijdag 28 oktober 09:00 - 10:30

Schrijf je **naam** en **studentnummer** op de regel hieronder.

Sla deze pagina niet om tot de surveillant vertelt dat het tentamen begonnen is.

Dit tentamen is een "gesloten boek"-tentamen. Echter, je mag gebruik maken van een tweezijdig beschreven of getypte pagina (A4-formaat). Verder is alleen het gebruik van een potlood of pen toegestaan.

Kladpapier is bijgesloten aan het eind van het tentamen. Je hoeft in je zelfgeschreven code geen "comments" te plaatsen, tenzij anders aangegeven. Geplaatste comments kunnen echter helpen bij eventuele gedeeltelijke puntentoekenning. In een geval van gebrek aan tijd kan geschreven pseudocode leiden tot een gedeeltelijke puntentoekenning.

Maximum aantal te behalen punten: **48**.

There can be only one.

Consider the C-program below. We want to print the result of the function unique.

```
#include <stdio.h>


#define ARR_LEN 7

int unique(int array[], int n)

int main(void)
{
     int array[ARR_LEN] = {1, 3, 5, 1, 7, 9, 3};

     printf("%i\n",unique(array, ARR_LEN));

     return 0;
}
```

0. (3 points) Only problem is, we do not have an implementation of the function `unique`. We need you to write it for us. What the function needs to do is to return the number of unique numbers in any given integer array. For example, in the array {2, 2, 4}, 2 unique numbers can be found (2 and 4).

```
int unique(int array[], int n)
{
```

There she goes.

Consider the C-program below, between whose lines are some numbers in between brackets.

```
#include <stdio.h>

void substitution(char*, char*);

int main(void)
{
    char name[] = "Kees Geel";
    char name2[] = "Coen Wauters";
    //(1)
    substitution(&name[0], &name2[0]);
    //(6)
    printf("%s\n%s\n", name, name2);

    return 0;
}

void substitution(char *a, char *b)
{
    //(2)
    char temp = *a;
    //(3)
    *a = *b;
    //(4)
    *b = temp;
    //(5)
}
```

1. (5 points) Suppose that each of the numbers represents a moment in time during this program's execution. For instance, if the program's execution is paused at number **(2)**, the value of **a** would be **0x33F1** and the value of **b** would be **0x33FB**. Record in the blank boxes below the values of this program's variables at each moment in time, whether in scope or not. Boxes for variables not (still) on the stack have been blacked out.

	name[0]	name2[0]	a	b	*a	*b	temp
(1)	'K'	'C'	■	■	■	■	■
(2)			0x33F1	0x33FB			■
(3)							
(4)							
(5)							
(6)			■	■	■	■	■

2. (1 point) And finally, what will be printed by the program on the previous page?

The stack is decked against you.

Consider the declaration and initialization (in C) of a stack for integers, below, wherein `CAPACITY` is some constant.

```
struct stack

{
     int size;
     int data[CAPACITY];
} stack;

stack.size = 0;
```

3. (3 points) Complete the implementation of **push()**, below, in such a way that the function pushes **x** at the top of the stack if not already at capacity. Upon succes, the function should return true. Upon failure (e.g., stack is at capacity), the function should return false.

```
bool push(int x)
{
```

4. (1 point) What is the time complexity (big O) of a the push operation for a **stack**?

Could not carry a note in a bucket.

5. (2 points) For searching purposes, it is ideal (and not likely to occur in practice), that every bucket in your hash table is filled up with a single element by a hash function. Why is that ideal?

6. (1 point) Within a hash table we can store elements inside buckets in different ways. For instance, we could use either an array or a linked list. What is the main difference between the two in terms of storage capacity?

A hedge between keeps friendships green.

Consider the following HMTL-document.

```
<!DOCTYPE html>

<html>
     <head>
          <style>




          </style>
     </head>
     <body>
          <p>Tentamen</p>
          <p>Programmeren 2</p>
          <p>Section HTML/CSS</p>
     </body>
</html>
```

7. (2 points) Using a CSS selector, change the color of all <p> elements to "green" by editing the code above.

8. (2 points) PHP is (frequently used as) an interpreted language, what do we mean by an **interpreted language**?

9. (3 points) Put the most appropriate language (PHP, SQL, HTML) next to corresponding letter of the program paradigm MVC. And what do the letters MVC stand for?

M____________:

V____________:

C____________:

May the force be with you.

```
<!DOCTYPE html>

<html>
     <body>
          <?php
          $staff = array(
               "Martijn" => "coordinator",
               "Hella" => "Head TA",
               "Wouter" => "Head TA"
          );
          foreach($staff as $x => $x_value) {
               echo "Name = " . $x . ", Role = " . $x_value;
               echo "<br>";
          }
          ?>
     </body>
</html>
```

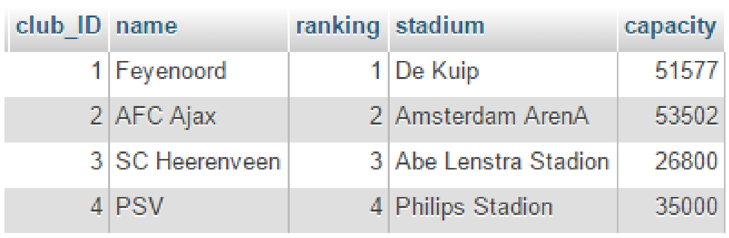
10. (2 points) What will be printed by the php-code above?

11. (1 point) What is the difference between the "==" and the "===" operator in JavaScript?

"Football is a simple game. Twenty-two men chase a ball for 90 minutes and at the end, the Germans always win."

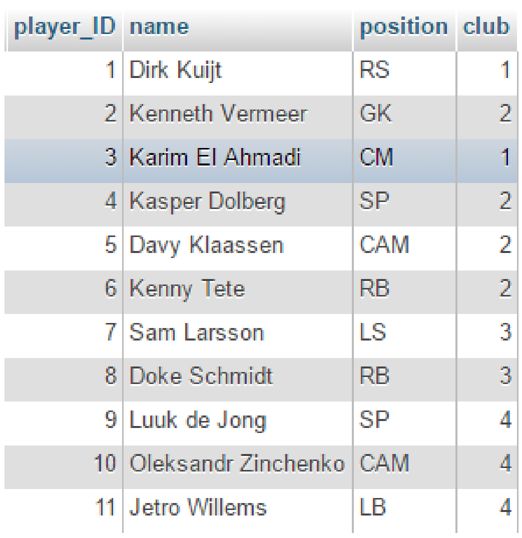
Consider the SQL-tables below from a database called `football_players`. On top is a table called `clubs`, wherein `club_ID` is a `Primary` key (with extra option `AUTO_INCREMENT`). On the bottom is a table called `players`, wherein `player_ID` is a `Primary` key (with extra option `AUTO_INCREMENT`) and `club` is a "foreign" key (whose values correspond to the values for `club_ID` in `clubs`).

[clubs]



club_ID	name	ranking	stadium	capacity
1	Feyenoord	1	De Kuip	51577
2	AFC Ajax	2	Amsterdam ArenA	53502
3	SC Heerenveen	3	Abe Lenstra Stadion	26800
4	PSV	4	Philips Stadion	35000

[players]



player_ID	name	position	club
1	Dirk Kuijt	RS	1
2	Kenneth Vermeer	GK	2
3	Karim El Ahmadi	CM	1
4	Kasper Dolberg	SP	2
5	Davy Klaassen	CAM	2
6	Kenny Tete	RB	2
7	Sam Larsson	LS	3
8	Doke Schmidt	RB	3
9	Luuk de Jong	SP	4
10	Oleksandr Zinchenko	CAM	4
11	Jetro Willems	LB	4

12. (2 points) Why is the column `rankings` in table `clubs` necessary to keep track of the league position of a club even though it has the same values stored as the values in column `club_ID`?

13. (1 point) Why does the column `player_ID` have to be unique in the table `players`, but not the column `club`?

14. (4 points) When we inserted the entries shown in the tables, we made 2 mistakes. First of all, the stadium in which "Feyenoord" plays its matches is not "De Kuip" (officially) but "Stadion Feijenoord". And secondly "Kenneth Vermeer" indeed used to play for "AFC Ajax" but nowadays he plays his matches for "Feyenoord". Create the two SQL queries needed to fix the mistakes.

```
UPDATE
```

```
UPDATE
```

15. (2 points) The KNVB (Koninklijke Nederlandse Voetbal Bond) wants to organize a European Cup final. Therefore they need to know which stadiums have a greater capacity than 50000. Create the SQL query with which they can retrieve a result set of rows, each of which includes the name of the `stadium` and the `capacity` of that stadium for all stadiums with a capacity greater than 50000.

```
SELECT
```

Because you're mine, you'll walk the line.

Consider the python program below.

```
a = 5; b = 2; c = 3
if a > b:
    a -= 2
    c /= 2
    if a < b:
        a*= 3
else:
    a = 9
```

16. (1 point) What value will **'a'** have after this code is run?

Consider the following for-loop in C.

```
for(int i = 0; i < 10; i++)
{
	printf("%i\n", i);
}
```

17. (1 point) In the space below, rewrite this loop to valid python.

One moment in time.

In C we can write a function to check whether a given character (b) has at least one occurrence in a given string (a), such a function would look something like the function below.

```
bool is_in_string(char *a, char b)
{
	for(int i = 0; i < strlen(a); i++)
	{
		if (a[i] == b)
		{
			return true;
		}
	}
	return false;
}
```

18. (2 points) Now write the same function in python in the space below.

```
class Athlete(object):
    type = 'Athlete'
    all_athletes = []

    def __init__(self, name, nationality='Blank'):
        self.name = name
        self.nationality = nationality
        self.all_athletes.append(name)

    def compose_print(self):
        return self.type + ', ' + self.name

    def __str__(self):
        return self.compose_print()


class Swimmer(Athlete):
    type = 'Swimmer'

    def __init__(self, name, nationality, distance):
        Athlete.__init__(self, name, nationality)
        self.distance = distance

    def compose_print(self):
        return self.type + ', ' + self.name + ', ' + \
                str(self.distance)


class Runner(Athlete):
    type = 'Runner'

    def __init__(self, name, nationality, distance):
        Athlete.__init__(self, name, nationality)
        self.distance = distance


class Sprinter(Runner):
    def __init__(self, name, nationality, distance):
        self.name = name
        self.nationality = nationality
        self.distance = distance


class Football_player(Athlete):
    type = 'Football_player'

    def __init__(self, name, nationality, club):
        Athlete.__init__(self, name, nationality)
        self.club = club
        self.all_athletes = []

    def change_club(self, new_club):
        self.club = new_club
```

Heritage does not equal destiny.

Consider the python code on the previous page, a file called `heritage.py`. Note that the special method `__str__` defines what will be printed by the object `o` when you call `print o`. You may assume that all the files created in this question (incl. heritage.py) are in the same directory.

Suppose we write the following program TT_001.py:

```
import heritage as h

print h.Swimmer('Ranomi Kromowidjojo', 'Dutch', '100m')
```

19. (2 points) What will be printed when we run this code?

Suppose we write the following program TT_002.py:

```
import heritage as h

print h.Sprinter('Usain Bolt', 'Jamaican', '100m')
```

20. (3 points) What will be printed when we run this code?

Suppose we write the following program TT_003.py:

```
import heritage as h

h.Swimmer('Ranomi Kromowidjojo', 'Dutch', '100m')
h.Football_player('Wayne Rooney', 'English', 'Man Utd')
h.Sprinter('Usain Bolt', 'Jamaican', '100m')
print h.Runner('Khalid Choukoud', 'Dutch', '42.195km').all_athletes
```

21. (4 points) What will be printed when we run this code?

Scrap paper.

Nothing on this page will be examined by the staff unless otherwise directed.