UNIVERSITEIT VAN AMSTERDAM 2018-2019

minor programmeren

Oefententamen Programmeren 1

versie: lente 2019

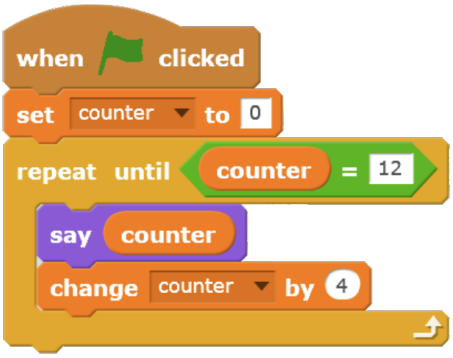
Vul hier je naam en studentnummer in vóór je begint:	/ 36 p

1. Je mag de vragen in Engels of Nederlands beantwoorden.
2. Dit is een "gesloten boek"-tentamen. Je mag voor het invullen je pen of potlood gebruiken, maar verder niets. Schrijf duidelijk en niet te groot.
3. Leg je studentenkaart (of ander ID met foto) klaar op je tafel. We komen langs om te kijken of je hierboven je naam hebt ingevuld en of deze klopt met je ID.
4. Laat het weten als je kladpapier nodig hebt.
5. Als je vragen hebt over hoe we iets bedoelen, dan kunnen we dat waarschijnlijk niet beantwoorden zonder een deel van het antwoord weg te geven (maar voel je vrij om het te proberen!).
6. Je hoeft geen comments in je code te schrijven.

Looping back to Scratch.

6 p **Question 1.**

Consider the Scratch script below.



In the space below, complete the translation of this Scratch script to a C program in such a way that its output is equivalent. (Your program's structure needn't be equivalent.) Assume that **counter** is an `int`, that **say** is `printf`, **set** means to assign a value to a variable, and **change** means to increment or decrement. Output `\n` after each line of text.

You may use any of the functions that you've learned in lectures or in the assignments. Pseudocode may earn you points, but usually less than correct C code!

```
#include <cs50.h>
#include <stdio.h>

int main(void)
{
```

Code reading and tracing.

In this section, questions use pseudocode like you have seen in the book.

1p **Question 2.**

Evaluate the following expressions:

a) `1.5 / 3 * 4`

b) `0.5 - 2 + 1.5`

c) `1.5 + 3 * 2`

d) `2 + 6 % 4`

1p **Question 3.**

Evaluate the following expressions:

a) `2 <= 1 || 3.0 <= 2.5`

b) `2 * 1.0 < 3 && !(3.0 <= 1.0)`

c) `1 >= 1 && 1 <= 2.5`

d) `!(2 <= 2.5) || 1 <= 2 % 3`

2p **Question 4.**

Write down the final value of each variable after running the following programs:

a)

```
fip = 3.0 / 1.5
dao = 1.0 - 1.5
toa = dao - 1
toa = fip * 2
toa = fip * 1
toa = toa + 2.0
```

b)

```
yrp = 1 / 2.0
eru = 1.0 - 1.5
if(yrp <= 2):
    yrp = yrp * 3
if(yrp == 1):
    eru = eru - 1.5
else:
    yrp = yrp + 3.0
```

2p **Question 5.**

Write down the final value of each variable after running the following programs:

a)

```
vug = 2
ing = 18
while(vug <= 10 && ing > 12)
    vug *= 2
    ing -= 3
```

b)

```
sal = ['u', 'r', 'u', 'b', 'a', 'd']
sal[5] = 'o'
for(mux = 2; mux <= 3; mux += 1)
  sal[mux] = sal[mux + 2]
```

2p **Question 6.**

Write down an algorithm that prints the first 256 values of the following sequence:

```
-1 3 7 11 15 19 23 ...
```

2p **Question 7.**

Write down an algorithm that prints the first 12 values of the following sequence:

```
3125 625 125 25 5 1 0 0 ...
```

2p **Question 8.**

Write down what is printed when the following program is run:

```
s = "onsvugingnon"
for(int oto = 3; oto < 9; oto += 1)
  print(s[oto + 2])
```

2p **Question 9.**

Rewrite this `for` loop as a `while` loop:

```
wah = 4
for(dip = 256; dip > 4; dip /= 4)
    wah += 3
```

2p **Question 10.**

Rewrite this `while` loop as a `for` loop:

```
aam = 4
while(aam <= 28)
    print(aam)
    aam += 9
```

A little code to write.

In this section, you earn more points by writing precise and correct C code.

6 p **Question 11.**

We would like to analyze large text files. To do this, we are writing a small library of functions, where each function performs a small analysis task. One of these is `count_vowels(string s)`, which should count how many vowels (a, e, i, o, u, y) occur in `s` and return that number. For example, in the string `"Hello, world!"`, there are three vowels. Implement that function below.

```
int count_vowels(string s)
{
```

6 p **Question 12.**

Another thing we're working on, is an application where (for some reason) users need to type characters, and each character is added to a string for further processing. Input is stopped whenever the user enters a `'Q'`. For example, when someone types `G<enter> T<enter> A<enter> C<enter> Q<enter>`, the array `input` should be filled with a string `"GTAC"`. Write the code to input all four letters using `get_char()` and fill the array.

```
// more than enough space
char input[10000];

// TODO
```

```
printf("You entered: %s\n", input);
```

Sort it out yourself.

Consider the following procedure (in pseudocode) for performing **insertion sort** on an array. It puts the numbers in an array in increasing order.

```
for i = 0 to n - 1
    element = array[i]
    j = i
    while(j > 0 and array[j-1] > element)
        array[j] = array[j-1]
        j = j - 1
    array[j] = element
    print array
```

2 p **Question 13.**

Perform the procedure on the array below, printing the array each time that we're at the bottom of the loop, like in the pseudocode above.

Remember to follow the procedure to the letter and to not take any shortcuts! Fill exactly as many rows as indicated by the print statement in the code.

(scribble version)

2	6	10	1	5

(final version)

2	6	10	1	5