CSE-A1111 Basic Course in Programming Y1. Exam 9.4.2014

Write the following information clearly on top of each paper you submit: name of the course, date of the exam, your full name, student ID, the total number of papers you submit, and your signature.

Important instructions: Use indentations of the length of two squares in your code. If your indentations are not clear enough, you lose points. You do not have to write any comments in your code. You can assume that the input given by the user is correct, if it is not told in the problem that you should handle the incorrect input. **Calculators and any extra material are not allowed. However, if your mother tongue is not Finnish, you may use a dictionary, if it does not contain any extra markings.**

Please fill in the course feedback form (you can find the link to it in the course Noppa page).

You do not have to justify your answers, the correct answer alone is enough. In questions 2-7 answer according to what the given code does, even if you think that the code works somehow oddly. Note that the points given from various questions vary from 2 to 25 points.

Question 1 (2 p)

If a dependent child is a person under 18 years of age who does not earn $10,000 or more a year, which expression would define a dependent child?

(a) `age < 18 and salary < 10000`
(b) `age < 18 or salary < 10000`
(c) `age <= 18 and salary <= 10000`
(d) `age <= 18 or salary <= 10000`

Question 2 (2 p)

What are the values of *girls*, *boys*, and *children* after the following code has been executed?

```
girls = 0
boys = 0
children = girls + boys
girls = 15
boys = 12
```

(a) 0, 0, 0
(b) 0, 0, 27
(c) 15, 12, 0
(d) 15, 12, 27

Question 3 (3 p)

What will be the value of the variable *z* after the following code is executed?

```
x = 1
y = 2
z = 3
if x < y:
    if y > 4:
        z = 5
    else:
        z = 6
```

Question 4 (3 p)
Consider the following block of code, where variables *a*, *b*, and *c* each store integer values:

```
if a > b:
   if b > c:
      answer = c
   else:
      answer = b
elif a > c:
   answer = c
else:
   answer = a
```

Which of the following sets of values for *a*, *b*, and *c* will cause *answer* to be assigned the value in variable *b*?

(a) a = 1, b = 2, c = 3
(b) a = 1, b = 3, c = 2
(c) a = 2, b = 1, c = 3
(d) a = 3, b = 2, c = 1

Question 5 (4 p)
What will be the value of *result* after the following code statements are executed?

```
nums1 = [1, -5, 2, 0, 4, 2, -3]
nums2 = [1, -5, 2, 4, 4, 2, 7]
result = 0
j = 0
while j < len(nums1):
   if nums1[j] != nums2[j]:
      result = result + 1
   j = j + 1
```

Question 6 (4 p)
What is the purpose or outcome of the following piece of code?

```
result = 0
for j in range(0, len(number)):
   if number[j] < 0:
      result = result + 1
```

(a) to find the smallest number in the list
(b) to count the negative numbers in the list
(c) to sum the negative numbers in the list
(d) to add 1 to each of the negative numbers in the list
(e) to find the index of the first negative number in the list

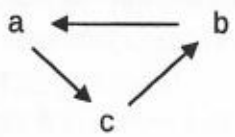
Question 7 (4 p)

What is the purpose or outcome of the following piece of code? Express your answer as a short phrase, like the phrases provided as possible answers in question 6.

```
result = 0
for count in range(1, num + 1):
   result = result + count
```

Question 8 (5 p)

There are three variables, *a*, *b* and *c*, which have been initialised to integer values. Write Python code to shift the values in these variables around so that *a* is given *b*'s original value, *b* is given *c*'s original value, and *c* is given *a*'s original value. The following diagram illustrates the direction of the shifts:



Question 9 (10 p)

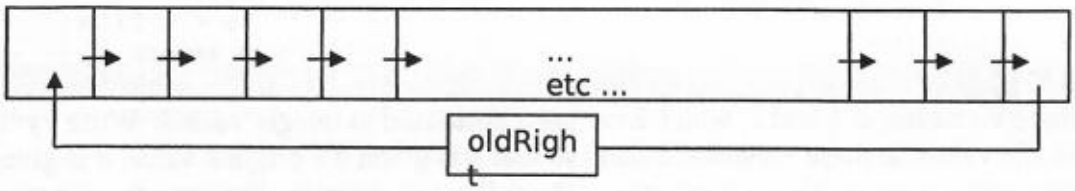
Three car rental places offer a similar car to be rented with various conditions: Car Rental A charges 50 euros per day and in addition to that 20 cnt per each driven kilometer. Car Rental B charges 150 euros per day, but no additional costs except the fuel. Car Rental C charges 100 per day. In the offer of Car Rental C, the customer can drive 1000 kilometers with that price during the whole rental period. If the total number of kilometers exceeds 1000, the customer pays 10 cnt for each extra kilometer. In all alternatives, the customer pays the fuel her/himself. Write a Python program which asks the user to input the number of days she/he wants to rent a car and the number of kilometers she/he is going to drive. The program calculates and outputs, which of the alternatives A, B and C is the cheapest for the user. The program also outputs the price of the cheapest alternative. If several alternatives have the same cheapest price, it is enough to give one of them (the program does not have to list them all).

Question 10 (12 p)

Write a Python function that will be given a list of integers as a parameter and will calculate and return (as a float) the average of all the integers in the list. You may assume that the list is not empty.

Question 11 (6 p)

We can represent a list of integers as a sequence of elements arranged from left to right, with the first element at the left and the last element at the right. Using this representation, a programmer wishes to move all elements of a list one place to the right, with the rightmost element being 'wrapped around' to the leftmost position, as shown in this diagram.



Here is the code that performs that shift for the list that the variable *values* refers to:

```
length = len(values)
oldRight = values[length - 1]
j = length - 1
while j > 0:
   values[j] = values[j - 1]
   j = j - 1
values[0] = oldRight
```

For example, if the list that *values* refers to initially contains the integers [1, 2, 3, 4, 5], once the code has executed it would contain [5, 1, 2, 3, 4].

Write Python code that will undo the effect of the above code. That is, write code that will move all the elements of the list one place to the left, with the leftmost element being wrapped around to the rightmost position.

Question 12 (20 p)

The results of two qualification rounds of a competition are stored in a text file such that each line contains the name of the competitor and the results of the rounds. The results are separated by colon (:). For example, lines of the files could be as follows:

```
Teemu Teekkari:160:240
Tiina Teekkari:250:80
Krista Kemisti:205:190
```

The competitor qualifies to the final, if he/she has at least 200 points from either Round 1 or Round 2 (or from the both rounds) and in addition he/she has at least total 360 points from the both rounds. (Note that the competitor has to fulfil the both conditions to qualify.) For example, in the case above Teemu and Krista qualify, but Tiina does not. Write a Python program which asks the user to input the name of the file which contains these results. The program reads the file and outputs the names and total points of those competitors who qualify.

The program must be able to handle the following errors:

- The file does not exist of it is not possible to read it because of some other reason.
- In some lines, the points cannot be converted to an integer.

In these cases, the program must output the type of the error occured and stop. The program does not have to continue reading after the defective line. You may also assume that each line of the file consists of three fields separated by a colon. Your program does not have to handle cases where the file contains empty lines of lines consisting of only one field, for example.

Question 13 (25 p)

Write a Python class `Passenger` to describe a member of the bonus program of an airline company. One `Passenger` object describes one member of the bonus program.

`Passenger` object should have the following data attibutes (fields):

- `__name` name of the passenger
- `__points` the number of bonus points the passenger currently has.

The class should have the following methods (unless otherwise told, the method does not have to return anything):

- `__init__(self, customer_name)` creates a new `Passenger` object. The name of the passenger is given as a parameter. The new passenger has 0 points.
- `get_name(self)` returns the name of the passenger
- `get_points(self)` returns the number of bonus points the passenger currently has.
- `add_points_from_flight(self, miles, business)` adds to the points the bonus points the passenger has earned from the flight whose information is given as parameters: `miles` gives the length of the flight (in miles) and the value of `business` is `True` for a flight in the business class and `False` for a flight in the economy class. For an economy class flight, the passenger is given one bonus point for every 10 miles (for example, the flight of 2456 miles gives 245 bonus points). For a business class flight, one bonus point is given for every 3 miles.
- `book_award_flight(self, miles, business)` decreases the number of the bonus points of the passenger by the number of the points needed to an award flight whose length and class are given as parameters (like in the previous method). For an economy class award flight, one bonus point is needed for each mile. For a business class award flight, two bonus points are needed for each mile. The method returns `True` if the passenger has enough bonus points for the award flight and `False` otherwise. In the latter case, the passenger cannot take the award flight and his/her bonus points are not decreased.
- `buy_reduced_ticket(self, price)` uses the bonus points to buy a flight ticket with a reduced price. The original price of the flight is given as a parameter. The reduction is 1 euro for each 100 bonus points the passenger has. The method decreases the number of the bonus points by the points used for reduction. The method always calculates the largest reduction which is possible with the existing bonus points. For example, if the passenger has 12450 bonus points and the original ticket price is 150 euros, the reduced price is 26 euros and the passenger has 50 points afterwards. However, at most as many bonus points are used as are needed to reduce the ticket price to 0 euros. If the passenger has no bonus points at all, the reduced price is the same as the original price. In all cases, the method returns the reduced price.
- `__str__(self)` returns a string containing the name and the number of bonus points.

In addition, write a main program which creates two `Passenger` objects and calls twice the method `add_points_from_flight` for both passengers. Then, the main program must call the method `book_award_flight` for the first passenger and output whether it was successful. After that the main program must call the method `buy_reduced_ticket` for the second passenger and output the reduced price. In the end, the program must output the names and the number of bonus points of the both passengers. You may decide the details of the passengers and the flights yourself. Your program does not have to ask anything from the user.