i

Informasjon

Exam in IN1000 autumn 2017

Time

8th December 09:00 (4 hours)

The lecturers will visit you around 10 o'clock.

Problems

Problems 2b and 2c are multiple choice questions. These will give the stated number of points if you answer correctly; a wrong answer or a missing answer will give you 0 points.

If you find a programming task unsufficiently defined or lacking information, make your own assumptions and describe these. You may also add functions or methods if needed. If you do, include a comment on what is added and why.

Permitted aids

Any written or printed material.

1

1a) 1 poeng

What is the value of **tall** when the following code has been executed?

```
tall = 3+1
tall = tall*2
```

Maximum marks: 1

2

1b) 1 poeng

What is the value of the variable **tekst** when the following code has been executed?

```
tekst = "a" + "c"
tekst = tekst + "b"
```

Maximum marks: 1

3 1c) 2 poeng

What is the value of the variable **j** when the following code has been executed?

```
i = 5
j = 10
while i<j:
  i = i+2
  j = j-1
```

Maximum marks: 2

4 1d) 2 poeng

What is printed on the screen when the following code is executed?

```
tallene = [1,6,4,2]
a = 0
b = 0
for tall in tallene:
  if tall<3:
    a = a+tall
  else:
    b = b+tall
print(a*b)
```

Maximum marks: 2

5 1e) 2 poeng

We have a function kalkuler shown below:

```
def kalkuler(tall):
  if tall<5:
    return tall*2
  else:
    return tall
```

What is printed on the screen when the following code is executed?

```
a = kalkuler(4+3)
print(a)
```

Maximum marks: 2

6 1f) 3 poeng

What is printed on the screen when the following code is executed?

```
class Tall:
  def __init__(self, a, b):
    self._a = a
    self._b = b
  def m1(self, c):
    self._b = self._b + c
  def m2(self):
    self._a = self._a + self._b
  def m3(self):
    return 2*self._a
t = Tall(3,2)
t.m1(1)
t.m2()
t.m2()
print(4+t.m3())
```

Maximum marks: 3

7 2a) 1 poeng

What is printed on the screen when the following code is executed?

```
a = [5, 2, 4]
b = a[0]
b = b + 1
print(a[0])
```

Maximum marks: 1

8

2b) 2 poeng

What is printed on the screen when the following code is executed?

```
a = [10, 8, 7]
b = a
c = b
b[0] = 3
c[1] = 4
print(a)
```

Select an alternative:

- [3,8,7]
- [3,4,7]
- [3,4]
- [10,4,7]
- [10,8,7]

Maximum marks: 2

9 2c) 3 poeng

Which four numbers are printed on the screen when the following code is executed?

```
class Person:
  def __init__(self, alder):
     self._alder = alder
  def doble_alder(self):
     self._alder = self._alder * 2
  def hent_alder(self):
     return self._alder

  def alder_som_maaneder(self):
     return self._alder * 12
p1 = Person(3)
p2 = p1
p3 = Person(5)
p4 = p3
p1.doble_alder()
print(p1.hent_alder())
print(p2.hent_alder())
print(p3.alder_som_maaneder())
print(p4.hent_alder())
```

Select an alternative:

- 6,3,60,60
- 3,3,60,5
- 6,3,60,5
- 6,6,60,60
- 6,6,60,5

Maximum marks: 3

10 3a) 5 poeng

Write a function **penger(femkroninger, kronestykker)** which takes as arguments a given number of 5 kr coins and 1 kr coins, respectively, and returns the total amount. For example, the call *penger(2,3)* shall return 13.

Fill in your answer here

Maximum marks: 5

11

3b) 5 poeng

A train company has a rule that children travel free when together with an adult. Write a function barnMedVoksen(alder1, alder2) which returns True if one of the parameter values is above (or equal to) 18 and the other is below 18. It doesn't matter which of the two ages is the adult or the child. Consequently, the call barnMedVoksen(18,5) shall return True, and the call barnMedVoksen(10,20) shall do the same. Furthermore, the call barnMedVoksen(20,30) shall return False.

Fill in your answer here

Maximum marks: 5

12

3c) 8 poeng

```
def allePositive(tallene) :
        for tall in tallene :
                if (tall < 0) :
                        return False
                return True
```

The function **allePositive** above is meant to check whether all values in the list **tallene** are positive. In this problem, we want you to:

1. describe the problem with the above solution
2. give an example of a list which does not produce the intended answer when used as an argument to the function
3. suggest improvements to make the function work as intended

Fill in your answer here

Format

Words: 0

Maximum marks: 8

13 3d) 8 poeng

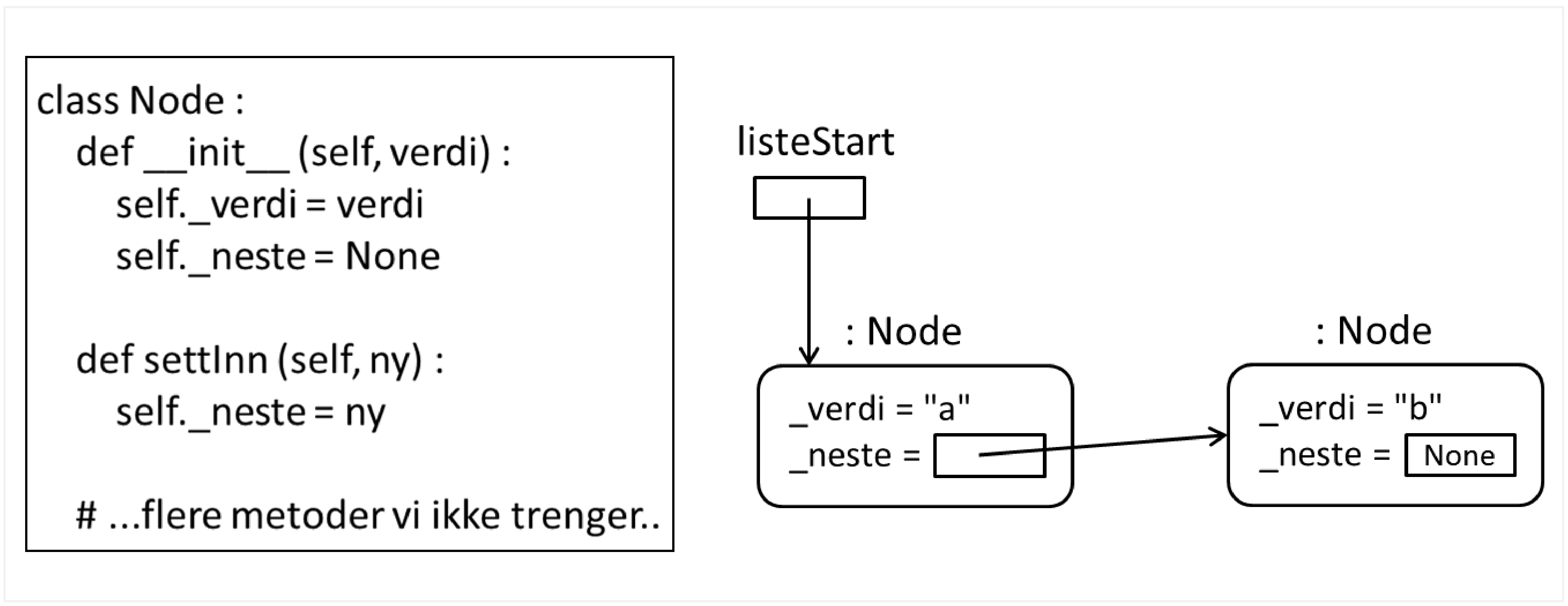
Write a function fyllTilTi(tallene) which accepts as parameter a list up to 10 numbers. (The length of the list will vary, but the maximum length is 10.) The function is to return a list of exactly 10 elements, in which all the numbers from the list *tallene* come first, and with sufficient elements of number 0 added to make the list 10 long.

Fill in your answer here

Maximum marks: 8

14

3e) 4 poeng



```
class Node :
   def __init__(self, verdi) :
      self._verdi = verdi
      self._neste = None

   def settInn (self, ny) :
      self._neste = ny

   # ...flere metoder vi ikke trenger..
```

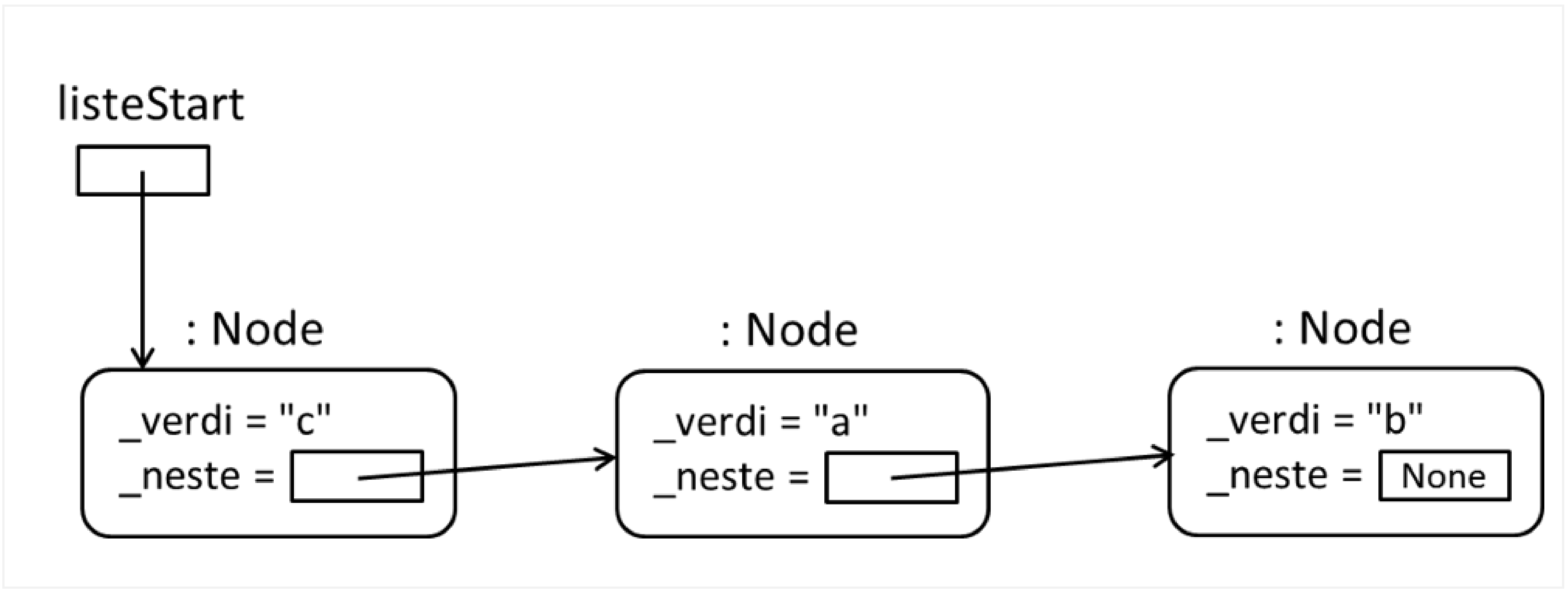
Given the class **Node** shown in the enclosed code, write a procedure **hovedprogram**() which creates 2 objects with values "a" and "b" in a structure as shown in the illustration. The variable **listeStart** (shown as a square) refers to (shown as an arrow) one of the objects. (The objects are shown as rectangles with rounded corners.) You may assume the the class **Node** has been imported to your program.

Fill in your answer here

Maximum marks: 4

15

3f) 3 poeng



Extend the procedure **hovedprogram** from problem e with code creating another object with value "c" and updates the references to obtain a data structure shown in the enclosed illustration.
The class **Node** remains unaltered, and objects of the class shall only be accessed through the given methods.

Fill in your answer here

Maximum marks: 3

16

4a) 8 poeng

This problem consists of several parts in which you are to implement component of a bigger system. If you skip a part, it is important that you read the whole text (which is enclosed in every problem part). In each part, you may use classes and methods from previous parts, even if you did not program them.

Write the class **Hytte** with methods as described in the document, problem a).

Fill in your answer here

Maximum marks: 8

17 4b) 10 poeng

Write the class Tur described in the document, problem b).
Fill in your answer here

```
1
```

Maximum marks: 10

18 4c) 10 poeng + 4d) 6 poeng

Write the class Turplanlegger as described in the document, problems c) and d).
Fill in your answer here

```
1
```

Maximum marks: 16

19 4e) 6 poeng

Write a test program as described in the documentation, problem e).

Fill in your answer here

Maximum marks: 6

20

5a) 10 poeng

In a company, each employee has a phone on which customers can contact them. When an employee goes out for lunch, he/she initiates a redirection to a colleague. This colleague may also have initiated a redirection, so a customer may be redirected several times before getting an answer. We can represent this as a list in which each employee is represented as an index from 0 and upwards. If an employee is available, the corresponding value in the list is -1. When an employee has redirected calls, the list value is the number to which the call is redirected. For example, the list [2, -1, -1, 0] represents a company with four employees; employee 0 has redirected to no 2, employees 1 and 2 are present, and employee 3 has redirected to no 0.

a) Write a function ring which takes a redirection list (as described above) and the employee number that the customer originally wanted to contact. The function shall return the employee who finally answers the call. In this problem, you can assume that a result is always possible (i.e., you will not enter an infinite loop). For example, the call ring(1, [2, -1, -1, 0]) shall return 1 (employee 1 is available), while the call ring(3, [2, -1, -1, 0]) shall return 2 (the call is redirected from no 3 to no 0 and then to no 2).

b) Write a function which takes a redirection list and checks whether there may exist an infinite loop of redirections in the list. One example is the list [1, -1, 3, 4, 2] in which a call to no 2 will be redirected to no 3, then to no 4 and then back to no 2 in a never-ending loop. If there is no such infinite loop of redirections, the function shall return True. If there exists at least one such infinite loop, it shall return False. Consequently, the call gyldig([2, -1, -1, 0]) shall return True, while the call gyldig([1, -1, 3, 4, 2]) skall return False.

Fill in your answer here

Maximum marks: 10

Question 16

Attached

Problem 4: Planning tours from lodge to lodge (40 points)

This problem consists of a number of parts of a larger system. If you skip a part, it is important that you read this entire text (a PDF file attached to each part). In each part, you can use the classes and methods from earlier parts, even if you have not programmed them.

You are to implement parts of a system that organises a variety of tours. Each tour lasts several days with accommodation in a new lodge every day. The part of the system you need to know consists of classes **Hytte**, **Tur** and **Turplanlegger**. Making a drawing of the structure for your own use is recommended.

Class **Turplanlegger** keeps track of all the lodges in the system with the help of a dictionary with lodge name as the key and the reference to the lodge as the value. Class **Turplanlegger** also has a list of references to tours.

In the class **Tur**, there is a text (which may contain blanks, but no line breaks) that describes the tour, and a list of references to the lodges frequented during the tour (all located in **Turplanlegger**'s dictionary).

Each object of class **Hytte** has a unique name, a number of beds and the price of accommodation per bed in this lodge. (Each bed in a lodge has the same price.)

Problem a) (8 points)

Write class **Hytte**. The constructor shall have parameters for the instance variables name, number of beds and the price for the accommodation. Moreover, the class has the following methods:

hentNavn returns the lodge’s name.

totPris returns total price for a number of people (the number is a parameter to the method) for one night.

skrivHytte prints a line on the screen with the lodge’s name, its number of beds and the price per bed. You need not worry about formatting.

sjekkPlass returns True if the lodge has enough beds for a number of people (parameter to the method), otherwise False.

Problem b) 10 points

Write the class **Tur** with a constructor that accepts a list of references to lodges and a (single-line) text that describes the trip. Moreover, the class has the following methods:

skrivTur prints on the screen a description of the tour, and then information about each lodge visited on the tour.

sjekkPrisPlass goes through all the lodges on the tour and checks if there are enough beds for a group of people in all the lodges, and if the total price for the tour (all persons in all lodges) is less than a maximum amount. The number of people and the maximum amount are parameters to the method. The method returns **True** if there is enough space and the cost is below the maximum amount, otherwise **False**.

Problem c) 10 points

Write the class **Turplanlegger** with the constructor and *one* of two methods: _**hytterFraFil** (to be written by you) and _ **turerFraFil** (should not be written by you, but may be used). The methods are called from the constructor and will read the input to the dictionary of lodges and the list of tours, respectively. The file names are parameters to the constructor and to each of these two methods.

Method _**hytterFraFil** opens and reads from the file whose name is given as a parameter to the method. The file has an unknown number of lines in the following format (two cabins shown)

Lodge_name   number_of_beds   price

Lodge_name   number_of_beds   price

:

where the first item to be read as a text and the last two items as integers and floating-point numbers. Each lodge is represented by an object of the class Hytte, and all Hytte objects are stored in a dictionary with the name of the lodge as a key. This dictionary is returned from the method when the file reading is through.

Method _**turerFraFil** opens and reads all information on all the tours in the system from the file whose name is given as a parameter, and creates and returns a list of **Tur** objects. You are not to write this, only use it in the constructor.

Problem d) 6 points

Write a method **finnTurer** in class **Turplanlegger** that can be used to identify the tours that have room for a given number of people in all the lodges, and where the total cost of all the people all days is below a maximum amount, and that do not last more than a maximum number of days. (The tours always go to a new lodge every day.) The method should go through all the tours, and print on the screen all information on the trips that meet the requirements. The number of people, the maximum amount and the maximum number of days are parameters to the method. You can use the methods from the previous problems regardless of whether you have written them.

Problem e) 6 points

Write a test program for the class **Turplanlegger** that does the following:

- Create an object of the class Turplanlegger that reads in data from files *hytter. txt* and *turer.txt.*
- Print all information about all tours that meet the following requirements:
  - does not last more than 5 days
  - has accommodation space on all the lodges for a group on 7 people
  - ... with an accommodation budget of 7000, for all 7 the whole trip.

Question 17
Attached

Problem 4: Planning tours from lodge to lodge (40 points)

This problem consists of a number of parts of a larger system. If you skip a part, it is important that you read this entire text (a PDF file attached to each part). In each part, you can use the classes and methods from earlier parts, even if you have not programmed them.

You are to implement parts of a system that organises a variety of tours. Each tour lasts several days with accommodation in a new lodge every day. The part of the system you need to know consists of classes **Hytte**, **Tur** and **Turplanlegger**. Making a drawing of the structure for your own use is recommended.

Class **Turplanlegger** keeps track of all the lodges in the system with the help of a dictionary with lodge name as the key and the reference to the lodge as the value. Class **Turplanlegger** also has a list of references to tours.

In the class **Tur**, there is a text (which may contain blanks, but no line breaks) that describes the tour, and a list of references to the lodges frequented during the tour (all located in **Turplanlegger**'s dictionary).

Each object of class **Hytte** has a unique name, a number of beds and the price of accommodation per bed in this lodge. (Each bed in a lodge has the same price.)

Problem a) (8 points)

Write class **Hytte**. The constructor shall have parameters for the instance variables name, number of beds and the price for the accommodation. Moreover, the class has the following methods:

hentNavn returns the lodge's name.

totPris returns total price for a number of people (the number is a parameter to the method) for one night.

skrivHytte prints a line on the screen with the lodge's name, its number of beds and the price per bed. You need not worry about formatting.

sjekkPlass returns True if the lodge has enough beds for a number of people (parameter to the method), otherwise False.

Problem b) 10 points

Write the class **Tur** with a constructor that accepts a list of references to lodges and a (single-line) text that describes the trip. Moreover, the class has the following methods:

skrivTur prints on the screen a description of the tour, and then information about each lodge visited on the tour.

sjekkPrisPlass goes through all the lodges on the tour and checks if there are enough beds for a group of people in all the lodges, and if the total price for the tour (all persons in all lodges) is less than a maximum amount. The number of people and the maximum amount are parameters to the method. The method returns **True** if there is enough space and the cost is below the maximum amount, otherwise **False**.

Problem c) 10 points

Write the class **Turplanlegger** with the constructor and *one* of two methods: **_hytterFraFil** (to be written by you) and **_ turerFraFil** (should not be written by you, but may be used). The methods are called from the constructor and will read the input to the dictionary of lodges and the list of tours, respectively. The file names are parameters to the constructor and to each of these two methods.

Method **_hytterFraFil** opens and reads from the file whose name is given as a parameter to the method. The file has an unknown number of lines in the following format (two cabins shown)

Lodge_name number_of_beds price

Lodge_name number_of_beds price

:

where the first item to be read as a text and the last two items as integers and floating-point numbers. Each lodge is represented by an object of the class Hytte, and all Hytte objects are stored in a dictionary with the name of the lodge as a key. This dictionary is returned from the method when the file reading is through.

Method **_turerFraFil** opens and reads all information on all the tours in the system from the file whose name is given as a parameter, and creates and returns a list of **Tur** objects. You are not to write this, only use it in the constructor.

Problem d) 6 points

Write a method **finnTurer** in class **Turplanlegger** that can be used to identify the tours that have room for a given number of people in all the lodges, and where the total cost of all the people all days is below a maximum amount, and that do not last more than a maximum number of days. (The tours always go to a new lodge every day.) The method should go through all the tours, and print on the screen all information on the trips that meet the requirements. The number of people, the maximum amount and the maximum number of days are parameters to the method. You can use the methods from the previous problems regardless of whether you have written them.

Problem e) 6 points

Write a test program for the class **Turplanlegger** that does the following:

- Create an object of the class Turplanlegger that reads in data from files *hytter. txt* and *turer.txt.*
- Print all information about all tours that meet the following requirements:
  - does not last more than 5 days
  - has accommodation space on all the lodges for a group on 7 people
  - ... with an accommodation budget of 7000, for all 7 the whole trip.

Question 18
Attached

Problem 4: Planning tours from lodge to lodge (40 points)

This problem consists of a number of parts of a larger system. If you skip a part, it is important that you read this entire text (a PDF file attached to each part). In each part, you can use the classes and methods from earlier parts, even if you have not programmed them.

You are to implement parts of a system that organises a variety of tours. Each tour lasts several days with accommodation in a new lodge every day. The part of the system you need to know consists of classes **Hytte**, **Tur** and **Turplanlegger**. Making a drawing of the structure for your own use is recommended.

Class **Turplanlegger** keeps track of all the lodges in the system with the help of a dictionary with lodge name as the key and the reference to the lodge as the value. Class **Turplanlegger** also has a list of references to tours.

In the class **Tur**, there is a text (which may contain blanks, but no line breaks) that describes the tour, and a list of references to the lodges frequented during the tour (all located in **Turplanlegger**'s dictionary).

Each object of class **Hytte** has a unique name, a number of beds and the price of accommodation per bed in this lodge. (Each bed in a lodge has the same price.)

Problem a) (8 points)

Write class **Hytte**. The constructor shall have parameters for the instance variables name, number of beds and the price for the accommodation. Moreover, the class has the following methods:

hentNavn returns the lodge's name.

totPris returns total price for a number of people (the number is a parameter to the method) for one night.

skrivHytte prints a line on the screen with the lodge's name, its number of beds and the price per bed. You need not worry about formatting.

sjekkPlass returns True if the lodge has enough beds for a number of people (parameter to the method), otherwise False.

Problem b) 10 points

Write the class **Tur** with a constructor that accepts a list of references to lodges and a (single-line) text that describes the trip. Moreover, the class has the following methods:

skrivTur prints on the screen a description of the tour, and then information about each lodge visited on the tour.

sjekkPrisPlass goes through all the lodges on the tour and checks if there are enough beds for a group of people in all the lodges, and if the total price for the tour (all persons in all lodges) is less than a maximum amount. The number of people and the maximum amount are parameters to the method. The method returns **True** if there is enough space and the cost is below the maximum amount, otherwise **False**.

Problem c) 10 points

Write the class **Turplanlegger** with the constructor and *one* of two methods: _**hytterFraFil** (to be written by you) and **_ turerFraFil** (should not be written by you, but may be used). The methods are called from the constructor and will read the input to the dictionary of lodges and the list of tours, respectively. The file names are parameters to the constructor and to each of these two methods.

Method _**hytterFraFil** opens and reads from the file whose name is given as a parameter to the method. The file has an unknown number of lines in the following format (two cabins shown)

Lodge_name   number_of_beds   price

Lodge_name   number_of_beds   price

:

where the first item to be read as a text and the last two items as integers and floating-point numbers. Each lodge is represented by an object of the class Hytte, and all Hytte objects are stored in a dictionary with the name of the lodge as a key. This dictionary is returned from the method when the file reading is through.

Method _**turerFraFil** opens and reads all information on all the tours in the system from the file whose name is given as a parameter, and creates and returns a list of **Tur** objects. You are not to write this, only use it in the constructor.

Problem d) 6 points

Write a method **finnTurer** in class **Turplanlegger** that can be used to identify the tours that have room for a given number of people in all the lodges, and where the total cost of all the people all days is below a maximum amount, and that do not last more than a maximum number of days. (The tours always go to a new lodge every day.) The method should go through all the tours, and print on the screen all information on the trips that meet the requirements. The number of people, the maximum amount and the maximum number of days are parameters to the method. You can use the methods from the previous problems regardless of whether you have written them.

Problem e) 6 points

Write a test program for the class **Turplanlegger** that does the following:

- Create an object of the class Turplanlegger that reads in data from files *hytter. txt* and *turer.txt.*
- Print all information about all tours that meet the following requirements:
    - does not last more than 5 days
    - has accommodation space on all the lodges for a group on 7 people
    - ... with an accommodation budget of 7000, for all 7 the whole trip.

Question 19
Attached

Problem 4: Planning tours from lodge to lodge (40 points)

This problem consists of a number of parts of a larger system. If you skip a part, it is important that you read this entire text (a PDF file attached to each part). In each part, you can use the classes and methods from earlier parts, even if you have not programmed them.

You are to implement parts of a system that organises a variety of tours. Each tour lasts several days with accommodation in a new lodge every day. The part of the system you need to know consists of classes **Hytte**, **Tur** and **Turplanlegger**. Making a drawing of the structure for your own use is recommended.

Class **Turplanlegger** keeps track of all the lodges in the system with the help of a dictionary with lodge name as the key and the reference to the lodge as the value. Class **Turplanlegger** also has a list of references to tours.

In the class **Tur**, there is a text (which may contain blanks, but no line breaks) that describes the tour, and a list of references to the lodges frequented during the tour (all located in **Turplanlegger**'s dictionary).

Each object of class **Hytte** has a unique name, a number of beds and the price of accommodation per bed in this lodge. (Each bed in a lodge has the same price.)

Problem a) (8 points)

Write class **Hytte**. The constructor shall have parameters for the instance variables name, number of beds and the price for the accommodation. Moreover, the class has the following methods:

hentNavn returns the lodge's name.

totPris returns total price for a number of people (the number is a parameter to the method) for one night.

skrivHytte prints a line on the screen with the lodge's name, its number of beds and the price per bed. You need not worry about formatting.

sjekkPlass returns True if the lodge has enough beds for a number of people (parameter to the method), otherwise False.

Problem b) 10 points

Write the class **Tur** with a constructor that accepts a list of references to lodges and a (single-line) text that describes the trip. Moreover, the class has the following methods:

skrivTur prints on the screen a description of the tour, and then information about each lodge visited on the tour.

sjekkPrisPlass goes through all the lodges on the tour and checks if there are enough beds for a group of people in all the lodges, and if the total price for the tour (all persons in all lodges) is less than a maximum amount. The number of people and the maximum amount are parameters to the method. The method returns **True** if there is enough space and the cost is below the maximum amount, otherwise **False**.

Problem c) 10 points

Write the class **Turplanlegger** with the constructor and *one* of two methods: _**hytterFraFil** (to be written by you) and **_ turerFraFil** (should not be written by you, but may be used). The methods are called from the constructor and will read the input to the dictionary of lodges and the list of tours, respectively. The file names are parameters to the constructor and to each of these two methods.

Method _**hytterFraFil** opens and reads from the file whose name is given as a parameter to the method. The file has an unknown number of lines in the following format (two cabins shown)

Lodge_name   number_of_beds   price

Lodge_name   number_of_beds   price

:

where the first item to be read as a text and the last two items as integers and floating-point numbers. Each lodge is represented by an object of the class Hytte, and all Hytte objects are stored in a dictionary with the name of the lodge as a key. This dictionary is returned from the method when the file reading is through.

Method _**turerFraFil** opens and reads all information on all the tours in the system from the file whose name is given as a parameter, and creates and returns a list of **Tur** objects. You are not to write this, only use it in the constructor.

Problem d) 6 points

Write a method **finnTurer** in class **Turplanlegger** that can be used to identify the tours that have room for a given number of people in all the lodges, and where the total cost of all the people all days is below a maximum amount, and that do not last more than a maximum number of days. (The tours always go to a new lodge every day.) The method should go through all the tours, and print on the screen all information on the trips that meet the requirements. The number of people, the maximum amount and the maximum number of days are parameters to the method. You can use the methods from the previous problems regardless of whether you have written them.

Problem e) 6 points

Write a test program for the class **Turplanlegger** that does the following:

- Create an object of the class Turplanlegger that reads in data from files *hytter. txt* and *turer.txt.*
- Print all information about all tours that meet the following requirements:
  - does not last more than 5 days
  - has accommodation space on all the lodges for a group on 7 people
  - ... with an accommodation budget of 7000, for all 7 the whole trip.