Name: Nico Kranni

Pair:

Amount of completed tasks:

Which tasks were left undone or incomplete?: Task 10 I dare to say is incomplete.

Self-assessment:

This exercise was easy than previous one. I might get back into flow.

Doing this exercise, I learned more about imports

I am still wondering…

I understood/did not understand that… ; I did/did not know that… ; I did/did not manage to do…

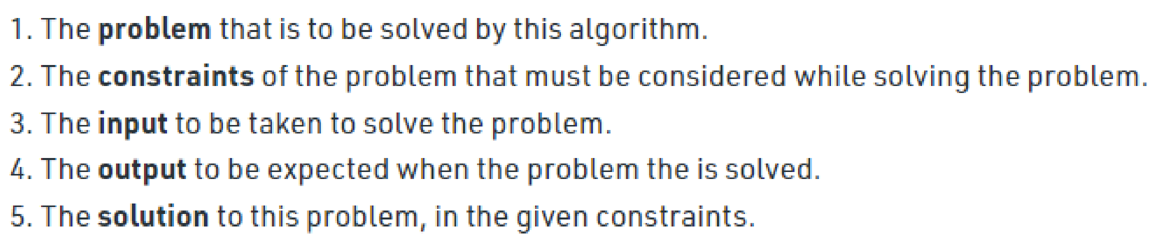
1.Explain the following terms:

a. Pseudocode

I believe sketch would describe it best. Sort of rough image of the final product. You use possibly use correct terms and names, but play around the idea of code

b. Algorithm

Set of rules or instructions to solve a specific problem. It works step by step to get desired results. There are certain rules that algorithm follows.



1. The **problem** that is to be solved by this algorithm.
2. The **constraints** of the problem that must be considered while solving the problem.
3. The **input** to be taken to solve the problem.
4. The **output** to be expected when the problem the is solved.
5. The **solution** to this problem, in the given constraints.

c. Data attribute

Either date, text or boolean. Numerals are considered measures. Data that describes other data.

Instance attribute belongs to only to one object, where class attribute is shared with all this class's objects.

d. Method

Function def, but method is class. Idea is similar. Method is called on object.

2. Take a look at the course’s assessment (points from exercises meaning certain grade). Write pseudocode for a program where user inputs the exercise points and program print out the grade.

>60 =0

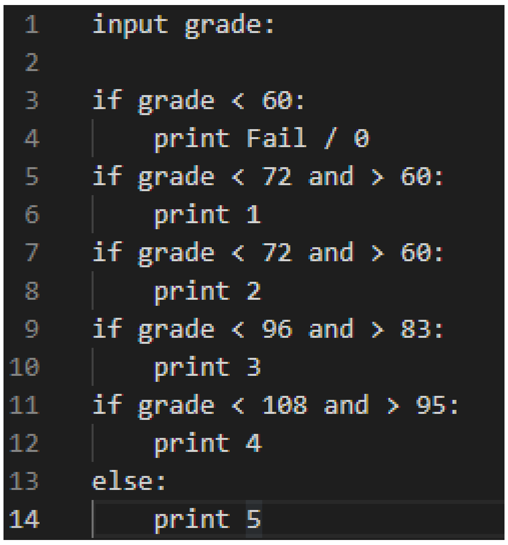
60-71=1

72-83=2

84-95=3

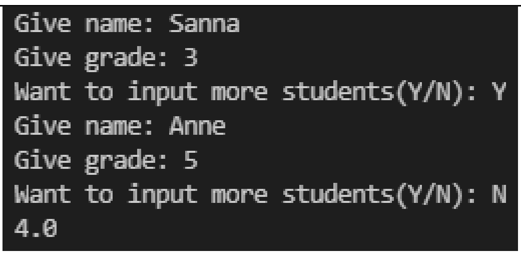
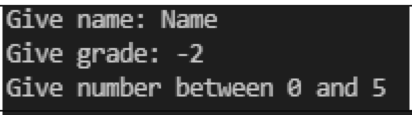
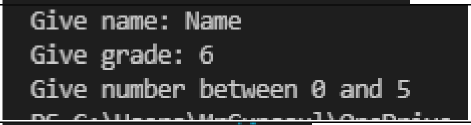
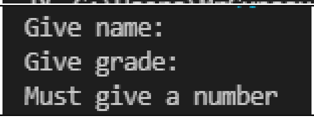
96-107=4

108=5



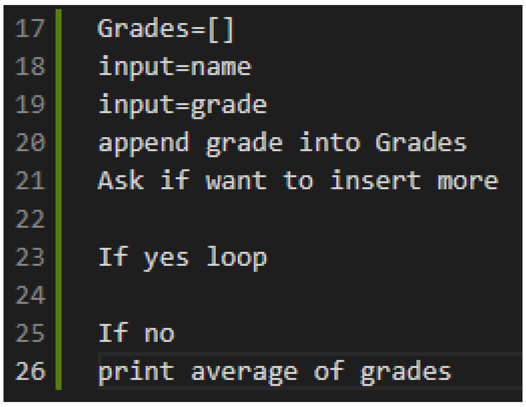
```
input grade:

if grade < 60:
    print Fail / 0
if grade < 72 and > 60:
    print 1
if grade < 72 and > 60:
    print 2
if grade < 96 and > 83:
    print 3
if grade < 108 and > 95:
    print 4
else:
    print 5
```

Test report

Task	**Input / action**	**Desired output**	**Actual output (use red color if desired output != actual output)**
3	User inputs integer out of range, e.g. -10 or 121.	Exercise points: -19 Error: exercise points cannot be < 0 or > 120.	points: -1 points have to be between 0 and 120
3	User inputs a valid integer, e.g. 78.	Exercise points: 78 Your grade is: 2	Points: 78 Grade: 2
3	<test every grade 0-5 that the points vs. grade works correctly>	Exercise points: Your grade is:	
5	User inputs 2 students. <You can also have some other way to ask the amount of entries than the one presented on the right. Just modify the desired output accordingly.>	Give name: Sanna Give grade: 3 Want to input more students (Y/N): Y Give name: Anne Give grade: 5 Want to input more students (Y/N): N Average grade of students is: 4	 Give name: Sanna Give grade: 3 Want to input more students(Y/N): Y Give name: Anne Give grade: 5 Want to input more students(Y/N): N 4.0
5	User inputs -2	Give name: *Name* Give grade: -2 Give number between 0 and 5	 Give name: Name Give grade: -2 Give number between 0 and 5
5	User inputs 6	Give name: *Name* Give grade: 6 Give number between 0 and 5	 Give name: Name Give grade: 6 Give number between 0 and 5
5	User inputs nothing	Give name: Give grade: Must give a number:	 Give name: Give grade: Must give a number
6	User runs the program <Run the program a couple of times so that you get each side up at least once>	This side is up: Heads Tossing the coin… Now this side is up: Tails	This side is up: Heads Tossing the coin… Now this side is up: Tails
7	User runs the program <Run the program a couple of times so that you get every side up at least once.>	This side is up: Heads Tossing the coin… Now this side is up: Coin defies gravity and disappeared.	This side up: Coin landed on the table upright! This side up: You buffoon! You threw the coin to the ground and now its gone! This side up: Why is coin flying up towards that weird looking hole….?

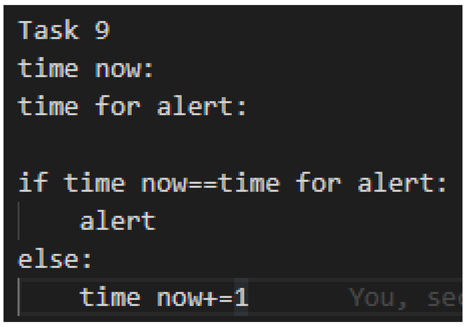
10	<Write test case depending on your implementation.>		Alarm works correctly with sound.

Task 4.



```
Grades=[]
input=name
input=grade
append grade into Grades
Ask if want to insert more

If yes loop

If no
print average of grades
```

Task 9.



```
Task 9
time now:
time for alert:

if time now==time for alert:
    alert
else:
    time now+=1         You, se
```

Code

Task 3

```
def grades():
    try:
        grade=int(input("points: "))
        if grade < 60 and grade >= 0:
            print ("Fail")
        if grade <= 71 and grade >= 60:
            print ("Grade: 1")
        if grade <=83 and grade >= 72:
            print ("Grade: 2")
        if grade <= 95 and grade >= 84:
            print ("Grade: 3")
        if grade <= 107 and grade >= 96:
            print ("Grade: 4")
        if grade >= 108:
            print ("Grade: 5")
        if grade < 0 or grade > 120:
            print("points have to be between 0 and 120")
    except ValueError:
        print("Value Error")

grades()
```

Task 5

```
gradeList=[]
def students():
    name=input("Give name: ")
    try:
        grade=int(input("Give grade: "))
        assert 0 <= grade < 6

        gradeList.append(grade)
        user=input("Want to input more students(Y/N): ")
        if user == 'Y':
            students()
        if user == 'N':
            print("Average of grades is: ",sum(gradeList)/len(gradeList))
    except AssertionError:
        print("Give number between 0 and 5")
    except ValueError:
        print("Must give a number")



students()
```

Task 10 (CODE IS NOT 100% MINE, ACTUAL SOURCE https://data-flair.training/blogs/alarm-clock-python/)

Code only edited by me.

```python
from tkinter import *
import datetime
import time
import winsound


def alarm(set_alarm_timer):
    while True:
        time.sleep(1)
        current_time = datetime.datetime.now()
        now = current_time.strftime("%H:%M:%S")
        # print(now)
        if now == set_alarm_timer:
            print("Time to Wake up")
            winsound.PlaySound("sound.wav", winsound.SND_ASYNC)
            break

# Gets information and sends them to actual function.
def actual_time():
    set_alarm_timer = f"{hour.get()}:{min.get()}:{sec.get()}"
    alarm(set_alarm_timer)

#Starts Tkinter, sets title and size
clock = Tk()
clock.title("DataFlair Alarm Clock")
clock.geometry("200x100")

# Text positions
Label(clock, text="Hour", font=60).place(x=0, y=20)
Label(clock, text="Min", font=60).place(x=75, y=20)
Label(clock, text="Sec", font=60).place(x=150, y=20)

# When to wake you up text.
Label(clock, text="When to wake you up", fg="blue", font=(
    "Helevetica", 7, "bold")).place(x=50, y=0)

# The Variables we require to set the alarm(initialization):
# Tkinter requiers different kind of variables, not regular python ones.
hour = StringVar()
min = StringVar()
sec = StringVar()

# Time required to set the alarm clock:
# We also assign these values to previous variables.
hourTime = Entry(clock, textvariable=hour, bg="pink",
                 width=10).place(x=0, y=40)
```

```
minTime = Entry(clock, textvariable=min, bg="pink",
                width=10).place(x=75, y=40)
secTime = Entry(clock, textvariable=sec, bg="pink",
                width=10).place(x=150, y=40)

# To take the time input by user:
submit = Button(clock, text="Set Alarm", fg="red", width=10,
                command=actual_time).place(x=100/2+10, y=75)
#Starts the window
clock.mainloop()
```