Question - 1:

Write a python program that plays a game “guess the number” as follows: The program chooses the number to be guessed by selecting an integer at random in the range 1 to 1000. The program then types:

- I have a number between 1 to 1000.
- Can you guess my number?
- Please type your first guess.

The player then types the first guess. The program responds with one of the following.

- Excellent!! You guessed the number!!! Would you like to play again (y or n)?
- Too high. Try again.
- Too low. Try again.

If the player’s guess is incorrect, you should loop (keep telling the player Too low or Too high) until the player finally gets the number right. Additionally, the program should count the number of guesses the player makes. If the player makes more than 10 guesses, then print “You should be able to do better!!!”, if he/she makes 10 guesses then print “Ahah! You know the secret!”, otherwise print “Either you know the secret or you got lucky!”

Run Program:

```
py oguz-aslanturk-dede-1.py
```

Sample User Input - 1:

```
I have a number between 1 to 1000.
Can you guess my number?
Please type your guess: 500
Too low. Try again.
Please type your guess: 900
Too high. Try again.
Please type your guess: 600
Too low. Try again.
```

Sample Output - 1:

```
Please type your guess: 709
Excellent!! You guessed the number!!!
Either you know the secret or you got lucky
```

Sample User Input - 2:

```
I have a number between 1 to 1000.
Can you guess my number?
Please type your first guess: 500
Too low. Try again.
```

```
Please type your guess: 750
Too high. Try again.
Please type your guess: 625
Too high. Try again.
Please type your guess: 570
Too high. Try again.
Please type your guess: 550
Too high. Try again.
Please type your guess: 525
Too high. Try again.
Please type your guess: 515
Too high. Try again.
Please type your guess: 507
Too low. Try again.
Please type your guess: 510
Too low. Try again.
Please type your guess: 513
Too high. Try again.
```

Sample Output - 2:

```
Please type your guess: 511
Excellent!! You guessed the number!!!
You should be able to do better!!!
```

Code:

```python
import random

def run():
   number_to_guess = random.randint(1, 1000)
   print("I have a number between 1 to 1000.")
   print("Can you guess my number?")

   result = False
   guess_count = 1

   is_first_guess = True
   while not result:
       result = check_guess(number_to_guess, is_first_guess)
       is_first_guess = False
       guess_count += 1

   print_result(guess_count)

def check_guess(number_to_guess, is_first_guess):
   user_guess = get_user_guess(is_first_guess)

   if number_to_guess == user_guess:
       print("Excellent!! You guessed the number!!!")
       return True
   else:
       if number_to_guess > user_guess:
           print("Too low. Try again.")
       else:
           print("Too high. Try again.")

       return False

def print_result(guess_count):
   if guess_count == 10:
       print("Ahah! You know the secret!")
   else:
       if guess_count > 10:
           print("You should be able to do better!!!")
       else:
           print("Either you know the secret or you got lucky")

def get_user_guess(is_first_guess):
   first_guess_text = ""

   while True:
       try:
           if is_first_guess: first_guess_text = "first "
           user_guess = int(input(f"Please type your 
{first_guess_text}guess: "))
       except ValueError:
           print("Please enter a valid integer 1-1000")
           continue
       if 1 <= user_guess <= 1000:
           return user_guess
       else:
           print('The integer must be in the range 1-1000')

run()
```

Question - 2:

A parking garage charges a $3.00 minimum fee to park for up to two hours. The garage charges an additional $1.00 per hour for each hour or part thereof in excess of two hours. The maximum charge for any given 24-hour period is $20.00. Assume that no car parks for longer than 24 hours at a time. Write a program that will read the hours parked for each customer who parked his/her car in this garage yesterday. Then calculate and display the parking charge for each car, the average parking hours for yesterday, and the total of yesterday's receipt. The program should use the function DetermineCharges to determine the charge for each customer. Assume that you don't know the number of customers in advance

Run Program:

```
py oguz-aslanturk-dede-2.py
```

Sample User Input - 1:

```
Please type parking hour (0 for exit): 20
Charge for 20 hour is : 20
```

Sample Output - 1:

```
Total receipt for yesterday is : 20
Average parking hour is : 20.00
```

Sample User Input - 2:

```
Please type parking hour (0 for exit): 5
Charge for 5 hour is : 6
```

Sample Output - 2:

```
Total receipt for yesterday is : 26
Average parking hour is : 12.50
```

Code:

```python
MIN_PARKING_HOUR = 2
UP_TO_TWO_HOUR_CHARGE = 3
PER_HOUR_CHARGE = 1
MAX_CHARGE = 20

def Run():
   charge_list = []
   parking_hour_list = []
   total_receipt = 0
   average_parking_hour = 0
```

```
        while True:
            parking_hour = GetParkingHour()

            if parking_hour == 0:
                PrintResult(total_receipt, average_parking_hour)
                return

            charge = DetermineCharges(parking_hour)
            print(f"Charge for {parking_hour} hour is : {charge}")

            parking_hour_list.append(parking_hour)
            charge_list.append(charge)

            total_receipt = TotalReceipt(charge_list)
            average_parking_hour =
CalculateAverageParkingHour(parking_hour_list)

            PrintResult(total_receipt, average_parking_hour)

    def DetermineCharges(hours_parked):
        if hours_parked <= MIN_PARKING_HOUR:
            charge = UP_TO_TWO_HOUR_CHARGE
        if hours_parked > MIN_PARKING_HOUR:
            charge = (hours_parked - MIN_PARKING_HOUR) * PER_HOUR_CHARGE +
UP_TO_TWO_HOUR_CHARGE

        if charge > MAX_CHARGE:
            return MAX_CHARGE
        else:
            return charge

    def GetParkingHour():
        while True:
            try:
                parking_hour = int(input("Please type parking hour (0 for
exit): "))
            except ValueError:
                print("Please enter a valid integer 0-24")
                continue
            if 0 <= parking_hour <= 24:
                return parking_hour
            else:
                print('Parking hour should be in the range 1-24')


    def TotalReceipt(charge_list):
        total_receipt = 0
        for charge in charge_list:
            total_receipt += charge

        return total_receipt

    def CalculateAverageParkingHour(parking_hour_list):
```

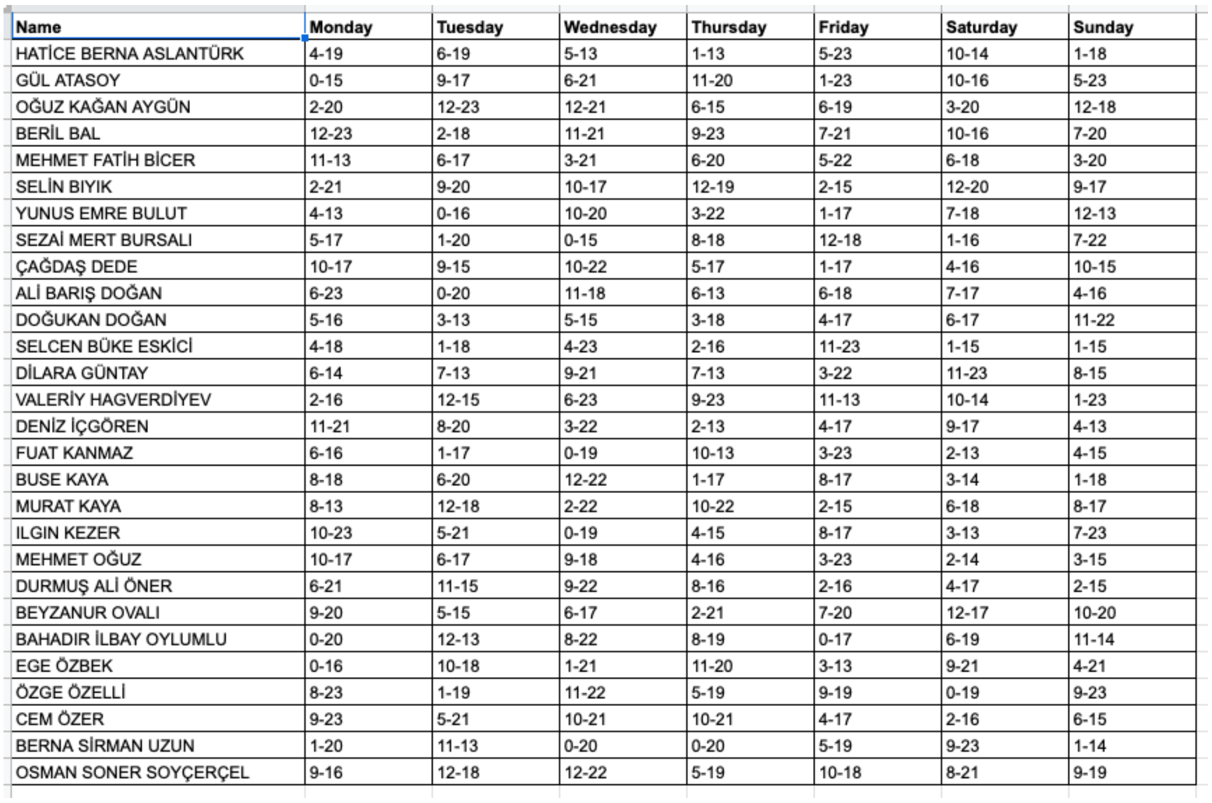
```python
    total_parkig_hour = 0
    for hour in parking_hour_list:
        total_parkig_hour += hour

    return total_parkig_hour / len(parking_hour_list)

def PrintResult(total_receipt, average_parking_hour):
    print("####### CUMULATIVE RESULT ######")
    print(f"Total receipt for yesterday is : {total_receipt}")
    print(f"Average parking hour is : {average_parking_hour:.2f}")
    print("################################")


Run()
```

Question - 3:

We have a company working with shifts, so they assign shifts to staff. Company also cares about any excuses staff have. So they do not assign a shift to one that has any excuse on that shift period.

Here we have an excuse list from a company(This list is downloaded as CSV and saved as txt). Program users choose a day and decide the shift beginning hour and end hour. Program finds available users to assign the shift.(Employee do not have any excuse in that time period).

This list includes staff full name, days of week and time period the user has an excuse so not available for shift. Time period is formatted as start of excuse and end hour of excuse in 24 hour format.



Name	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
HATİCE BERNA ASLANTÜRK	4-19	6-19	5-13	1-13	5-23	10-14	1-18
GÜL ATASOY	0-15	9-17	6-21	11-20	1-23	10-16	5-23
OĞUZ KAĞAN AYGÜN	2-20	12-23	12-21	6-15	6-19	3-20	12-18
BERİL BAL	12-23	2-18	11-21	9-23	7-21	10-16	7-20
MEHMET FATİH BİCER	11-13	6-17	3-21	6-20	5-22	6-18	3-20
SELİN BIYIK	2-21	9-20	10-17	12-19	2-15	12-20	9-17
YUNUS EMRE BULUT	4-13	0-16	10-20	3-22	1-17	7-18	12-13
SEZAİ MERT BURSALI	5-17	1-20	0-15	8-18	12-18	1-16	7-22
ÇAĞDAŞ DEDE	10-17	9-15	10-22	5-17	1-17	4-16	10-15
ALİ BARIŞ DOĞAN	6-23	0-20	11-18	6-13	6-18	7-17	4-16
DOĞUKAN DOĞAN	5-16	3-13	5-15	3-18	4-17	6-17	11-22
SELCEN BÜKE ESKİCİ	4-18	1-18	4-23	2-16	11-23	1-15	1-15
DİLARA GÜNTAY	6-14	7-13	9-21	7-13	3-22	11-23	8-15
VALERİY HAGVERDİYEV	2-16	12-15	6-23	9-23	11-13	10-14	1-23
DENİZ İÇGÖREN	11-21	8-20	3-22	2-13	4-17	9-17	4-13
FUAT KANMAZ	6-16	1-17	0-19	10-13	3-23	2-13	4-15
BUSE KAYA	8-18	6-20	12-22	1-17	8-17	3-14	1-18
MURAT KAYA	8-13	12-18	2-22	10-22	2-15	6-18	8-17
ILGIN KEZER	10-23	5-21	0-19	4-15	8-17	3-13	7-23
MEHMET OĞUZ	10-17	6-17	9-18	4-16	3-23	2-14	3-15
DURMUŞ ALİ ÖNER	6-21	11-15	9-22	8-16	2-16	4-17	2-15
BEYZANUR OVALI	9-20	5-15	6-17	2-21	7-20	12-17	10-20
BAHADIR İLBAY OYLUMLU	0-20	12-13	8-22	8-19	0-17	6-19	11-14
EGE ÖZBEK	0-16	10-18	1-21	11-20	3-13	9-21	4-21
ÖZGE ÖZELLİ	8-23	1-19	11-22	5-19	9-19	0-19	9-23
CEM ÖZER	9-23	5-21	10-21	10-21	4-17	2-16	6-15
BERNA SİRMAN UZUN	1-20	11-13	0-20	0-20	5-19	9-23	1-14
OSMAN SONER SOYÇERÇEL	9-16	12-18	12-22	5-19	10-18	8-21	9-19

Run Program:

```
py oguz-aslanturk-dede-3.py
```

Sample User Input - 1:

```
Monday : 1
Tuesday : 2
Wednesday : 3
Thursday : 4
Friday : 5
Saturday : 6
Sunday : 7
Exit : 0
Please type shift day: 1
Please type shift beginning hour: 16
Please type shift end hour: 21
```

Sample Output - 1:

```
Available personnel for this shift:
GÜL ATASOY
MEHMET FATİH BİCER
YUNUS EMRE BULUT
DOĞUKAN DOĞAN
DİLARA GÜNTAY
VALERİY HAGVERDİYEV
FUAT KANMAZ
MURAT KAYA
EGE ÖZBEK
OSMAN SONER SOYÇERÇEL
OSMAN SONER SOYÇERÇEL
```

Sample User Input - 2:

```
Monday : 1
Tuesday : 2
Wednesday : 3
Thursday : 4
Friday : 5
Saturday : 6
Sunday : 7
Exit : 0
Please type shift day: 5
Please type shift beginning hour: 9
Please type shift end hour: 12
```

Sample Output - 2:

Available personnel for this shift:
SEZAİ MERT BURSALI

Code:

```
def run():
   excuse_file = open('personnel_excuse_list.txt', 'r')
   line = excuse_file.readline()
   excuse_list = []

   while line != '':
       line = excuse_file.readline()
       if line != '':
           personnel_excuses = line.split(',')
       excuse_list.append(personnel_excuses)

   excuse_file.close()

   day = get_working_day()
   if day == 0:
       return

   working_hour_begins = get_shift_start()
   working_hour_ends = get_shift_end(working_hour_begins)

   available_personnel = find_available_personnel(excuse_list, day,
working_hour_begins, working_hour_ends)
   print_result(available_personnel)


def get_working_day():
   print_days()
   while True:
       try:
           shift_day = int(input("Please type shift day: "))
       except ValueError:
           print("Please enter a valid day 0-7")
           continue
       if 0 <= shift_day <= 7:
           return shift_day
       else:
           print('Shift day should be in the range 1-7')

def get_shift_start():
   while True:
       try:
           shift_start = int(input("Please type shift beginning hour:
"))
       except ValueError:
           print("Please enter a valid a valid shift 0-23")
           continue
       if 0 <= shift_start <= 23:
           return shift_start
       else:
           print('Shift start should be in the range 0-23')

def get_shift_end(working_hour_begins):
   while True:
       try:
           shift_end = int(input("Please type shift end hour: "))
       except ValueError:
           print(f"Please enter a valid shift end hour
{working_hour_begins}-23")
           continue
       if working_hour_begins < shift_end <= 23:
           return shift_end
       else:
           print(f'Shift end should be in the range
{working_hour_begins}-23')

def print_result(available_personnel):
   print("Available personnel for this shift: ")

   for personnel in available_personnel:
       print(personnel)

def print_days():
   print("Monday : 1")
   print("Tuesday : 2")
   print("Wednesday : 3")
   print("Thursday : 4")
   print("Friday : 5")
   print("Saturday : 6")
   print("Sunday : 7")
   print("Exit : 0")

def find_available_personnel(excuse_list, day, working_hour_begins,
working_hour_ends):
   available_personnel = []

   for personnel_excuse_list in excuse_list:
       if check_personnel_status(personnel_excuse_list[day],
working_hour_begins, working_hour_ends ):
           available_personnel.append(personnel_excuse_list[0])

   return available_personnel

def check_personnel_status(personnel_excuse_hours,
working_hour_begins, working_hour_ends):
   hours = personnel_excuse_hours.split('-')
   excuse_hour_begins = int(hours[0])
   excuse_hour_ends = int(hours[1])

   if excuse_hour_ends <= working_hour_begins:
        return True
    if working_hour_ends <= excuse_hour_begins:
        return True

    return False

run()
```