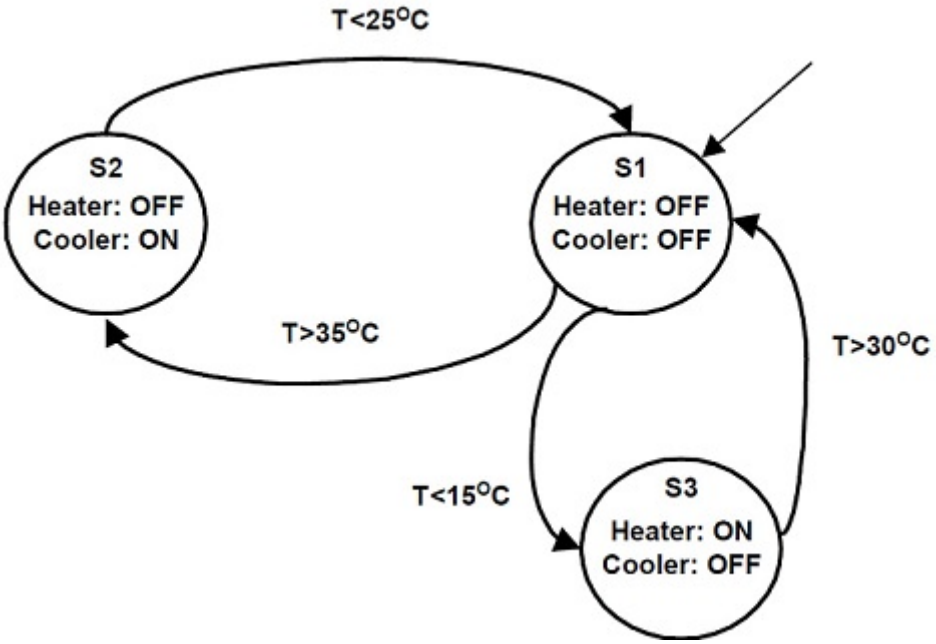
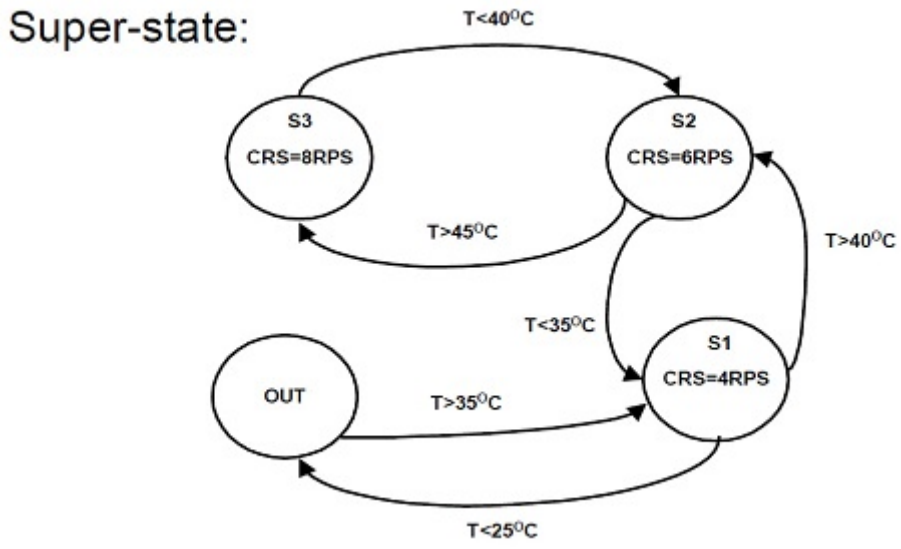
سوال ۱

طراحی و پیاده سازی یک سامانه Conditioning Air با دو Super-State با زبان برنامه نویسی دلخواه و ارائه یک گزارش از روند پیاده‌سازی. توجه کنید که برای فعال کردن heater بایستی سه حالت کاری و برای فعال کردن cooler سه حالت کاری در نظر بگیرید.

جواب سوال ۱





توضیح منطق انکوباتور

الف) **وضعیت‌ها (States)**:

- S1: وضعیت اولیه
- S2: وضعیت دومی
- S3: وضعیت سومی

ب) **منطق وضعیت‌ها**:

- اگر وضعیت فعلی (PS) برابر با S1 باشد:
  - اگر دما (T) بیشتر از ۳۵ باشد، وضعیت بعدی (NS) به S2 تغییر می‌کند و وضعیت زیرین S2 به S1 تنظیم می‌شود.
  - اگر دما کمتر از ۱۵ باشد، وضعیت بعدی به S3 تغییر می‌کند و وضعیت زیرین S3 به S1 تنظیم می‌شود.
  - در غیر این صورت، وضعیت بعدی همچنان S1 است.
- اگر وضعیت فعلی برابر با S2 باشد:
  - وضعیت زیرین بر اساس وضعیت فعلی S2 و دما تعیین می‌شود.
  - اگر دما کمتر از ۲۵ باشد، وضعیت بعدی به S1 تغییر می‌کند و وضعیت زیرین S2 ریست می‌شود.
- اگر وضعیت فعلی برابر با S3 باشد:
  - وضعیت زیرین بر اساس وضعیت فعلی S3 و دما تعیین می‌شود.
  - اگر دما بیشتر از ۳۰ باشد، وضعیت بعدی به S1 تغییر می‌کند و وضعیت زیرین S3 ریست می‌شود.

ج) **ورودی‌ها (Inputs)**:

- T: دما ـ یک عدد صحیح میان ۰ تا ۱۰۰.
- PS: وضعیت فعلی ـ می‌تواند یکی از وضعیت‌های S1، S2، یا S3 باشد.

د) **خروجی‌ها (Outputs)**:

- NS: وضعیت بعدی ـ یکی از وضعیت‌های S1، S2، یا S3.
- Error: پیام خطا در صورت وجود.

ه) **تست (Testing)**:

- برای تست کردن سیستم، از یک سری دماها با مقادیر مشخص استفاده می‌شود تا واکنش سیستم به هر دما مشاهده شود.
- خروجی انتظاری و خروجی فعلی مقایسه می‌شود.
- در صورت وجود هر گونه ناهمخوانی بین خروجی مورد انتظار و خروجی فعلی، مشکلی در منطق انکوباتور وجود دارد و باید بازبینی شود.

و) **چرخه اصلی**:

- یک لیست از دماها با مقادیر تصادفی ایجاد می‌شود.
- برای هر دما، تابع `encubate` فراخوانی می‌شود تا وضعیت بعدی انکوباتور را تعیین کند.
- این فرآیند تا زمانی که وضعیت استقرار پیدا کند (یعنی تغییرات وضعیت متوقف شود) ادامه می‌یابد.
- در نهایت، وضعیت استقرار یافته چاپ می‌شود.

ز) **مدیریت خطا**:

- در صورتی که وضعیت فعلی یا وضعیت زیرین به یک مقدار غیر معتبر تغییر کند، یک خطای "Invalid state" رخ می‌دهد.

ورودی‌ها و مقادیر اولیه

الف) در ابتدا، بسته‌ی مربوط به enum ورودی گرفته می‌شود. این بسته برای تعریف مقادیر ثابت در پایتون استفاده می‌شود.

ب) دو کلاس با نام‌های MainState و SubState تعریف شده‌اند. هر دوی این کلاس‌ها از نوع Enum هستند و به ترتیب برای نگه‌داری وضعیت‌های اصلی و زیرین انکوباتور مورد استفاده قرار می‌گیرند.

کلاس انکوباتور

الف) کلاس Encubator به عنوان کلاس اصلی و مرکزی برنامه تعریف شده‌است.

ب) درون این کلاس، منطق مربوط به وضعیت‌ها، پردازش ورودی دما، و تصمیم‌گیری بر اساس وضعیت فعلی و دما جاری قرار دارد.

محیط اصلی برنامه

الف) در قسمت __main__، یک نمونه از کلاس Encubator ساخته می‌شود.

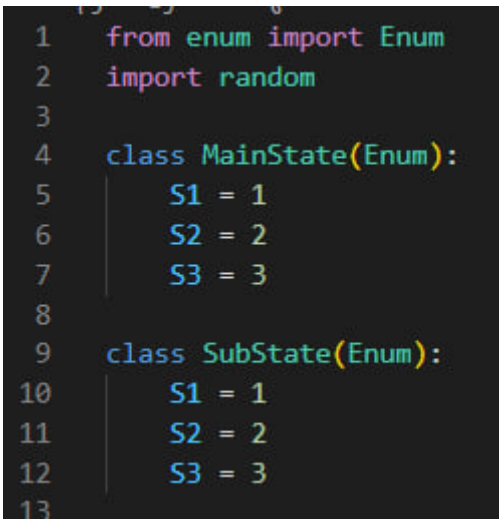
ب) با استفاده از مقادیر دماهای تصادفی، واکنش انکوباتور به دماهای مختلف مورد آزمون و بررسی قرار می‌گیرد.

ج) همچنین این قسمت می‌تواند برای نمایش وضعیت‌ها و تغییرات آنها استفاده شود.

نتایج اجرای برنامه

برنامه در هر مرحله از اجرا، دما و وضعیت مرتبط با آن را نشان می‌دهد. در زیر خروجی‌های مختلف برنامه نمایش داده شده‌است:

```
Temperature: 3
State: S3
S3 State: S2
...
(Other Outputs)
...
Temperature: 40
State: S2
S2 State: S2
```



```python
from enum import Enum
import random

class MainState(Enum):
    S1 = 1
    S2 = 2
    S3 = 3

class SubState(Enum):
    S1 = 1
    S2 = 2
    S3 = 3
```

شکل ۱: این بخش مربوط به تعریف ورودی‌ها و انواع متغیرهای شمارشی (Enum) است. دو Enum به نام‌های MainState و SubState تعریف شده‌اند که هر کدام سه حالت (S1, S2, S3) دارند.

```python
class Encubator:
    Codeium: Refactor | Explain | Generate Docstring
    def __init__(self):
        self.PS = MainState.S1
        self.S2_PS = SubState.S1
        self.S3_PS = SubState.S1

    Codeium: Refactor | Explain | Generate Docstring
    def process_temperature(self, T: int):
        if self.PS == MainState.S1:
            if T > 35:
                self.PS = MainState.S2
            elif T < 15:
                self.PS = MainState.S3

        elif self.PS == MainState.S2:
            if T < 25:
                self.PS = MainState.S1
            else:
                if self.S2_PS == SubState.S1:
                    if T > 40:
                        self.S2_PS = SubState.S2
                elif self.S2_PS == SubState.S2:
                    if T > 45:
                        self.S2_PS = SubState.S3
                    elif T < 35:
                        self.S2_PS = SubState.S1
                elif self.S2_PS == SubState.S3:
                    if T < 40:
                        self.S2_PS = SubState.S2

```

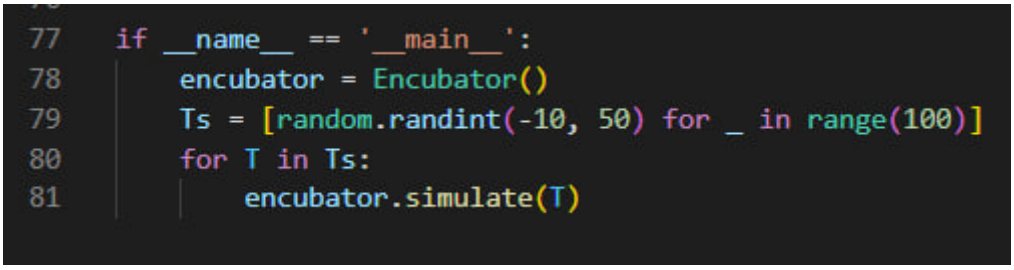
شکل ۲: اینجا یک کلاس به نام `Encubator` تعریف شده است که متدها و ویژگی‌های اولیه‌ی آن در این قسمت آورده شده. متد `process_temperature` بر اساس دمای ورودی حالت‌های مختلف را مدیریت می‌کند و بر اساس دما، حالت‌های اصلی و زیر حالت‌ها را تغییر می‌دهد.

```
43          elif self.PS == MainState.S3:
44              if T > 30:
45                  self.PS = MainState.S1
46              else:
47                  if self.S3_PS == SubState.S1:
48                      if T < 5:
49                          self.S3_PS = SubState.S2
50                  elif self.S3_PS == SubState.S2:
51                      if T < -5:
52                          self.S3_PS = SubState.S3
53                      elif T > 15:
54                          self.S3_PS = SubState.S1
55                  elif self.S3_PS == SubState.S3:
56                      if T > 5:
57                          self.S3_PS = SubState.S2
58
59          else:
60              raise Exception("Invalid Main State")
61
```

شکل ۳: این قسمت ادامه‌ی تابع `process_temperature` است و به مدیریت حالت S3 می‌پردازد. اگر دما یا حالتی خارج از حالت‌های تعریف شده باشد، یک خطا به کاربر نشان داده می‌شود.

```
def simulate(self, T: int):
    while True:
        prev_PS = self.PS
        prev_S2_PS = self.S2_PS
        prev_S3_PS = self.S3_PS
        self.process_temperature(T)
        if prev_PS == self.PS and prev_S2_PS == self.S2_PS and prev_S3_PS == self.S3_PS:
            break
    print(f'Temperature: {T}')
    print('State:', self.PS.name)
    if self.PS == MainState.S2:
        print('S2 State:', self.S2_PS.name)
    elif self.PS == MainState.S3:
        print('S3 State:', self.S3_PS.name)
```

شکل ۴: متد `simulate` تعریف شده است که وظیفه‌ی آن شبیه‌سازی واکنش حالت‌ها به دما است. در این متد، تا زمانی که هیچ تغییری در حالت‌ها ایجاد نشود، پردازش دما ادامه می‌یابد. در نهایت، دما و حالت‌های فعلی نمایش داده می‌شوند.



```
77  if __name__ == '__main__':
78      encubator = Encubator()
79      Ts = [random.randint(-10, 50) for _ in range(100)]
80      for T in Ts:
81          encubator.simulate(T)
```

شکل ۵: این بخش کد اصلی برنامه است که یک نمونه از کلاس Encubator ایجاد می‌کند و سپس ۱۰۰ دمای تصادفی می‌سازد و برای هر دما، شبیه‌سازی را اجرا می‌کند.