In [1]:

```
from dsLecture3 import ArrayQueue
```

In [2]:

```
class Task:
    def __init__(self, list_of_jobs):
        self._jobs = ArrayQueue()
        for job in list_of_jobs:
            self._jobs.enqueue(job)

    def get_next_job(self):
        if self._jobs.is_empty():
            return None
        return self._jobs.dequeue()

    def has_more_job(self):
        return not self._jobs.is_empty()

class RR_Scheduler:
    # q: a queue of task objects

    def __init__(self):
        self._q = ArrayQueue()

    def feed_task(self, task):   # Task: a list of tasks that requires a unit operation time
        self._q.enqueue(task)

    def get_next_job(self):
        if self._q.is_empty():
            return None

        next_task = self._q.dequeue()
        next_job = next_task.get_next_job()

        if next_task.has_more_job():
            self._q.enqueue(next_task)                  # Round-Robin Scheduling

        return next_job
```

In [3]:

```
t1 = Task(['1A', '1B', '1C', '1D'])
t2 = Task(['2A', '2B', '2C', '2D'])
t3 = Task(['3A', '3B', '3C', '3D'])
t4 = Task(['4A', '4B', '4C', '4D'])

sch = RR_Scheduler()
sch.feed_task(t1)
print(sch.get_next_job())
sch.feed_task(t2)
print(sch.get_next_job())
print(sch.get_next_job())
print(sch.get_next_job())
sch.feed_task(t3)
print(sch.get_next_job())
sch.feed_task(t4)
print(sch.get_next_job())

while True:
    next_job=sch.get_next_job()
    if next_job == None:
        break

    print(next_job)
```

```
1A
1B
2A
1C
2B
```

2B
1D
3A
2C
4A
3B
2D
4B
3C
4C
3D
4D

In [ ]:

```
```

In [ ]:

```
```