In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
import random as r

class People:
    def __init__(self, num):
        self.number = num
        self.recovered = False
        self.daysSick = 0
```

In [2]:

```
class Outbreak:
    def __init__(self, recoveryTime, population, init_deathrate, infect_rate, recover_rate, name=""):
        self.rect  = recoveryTime
        self.population = population
        self.deathrate  = init_deathrate
        self.infrate  = infect_rate
        self.recrate  = recover_rate
        if name != "": self.name = name

        self.totInfect = 0
        self.totDead   = 0
        self.totRec    = 0

        self.clean     = []
        for i in range(population): self.clean.append(People(i))
        self.infected  = []

        self.infectCt  = []
        self.deathCt   = []
        self.recCt     = []

        self.pmrate       = []
        self.prrate       = []
        self.cmrate       = []
        self.crrate       = []

    def info(self):
        print ("Name:",self.name)
        print ("Deathrate:", self.deathrate)
        print ("Infectrate:", self.infrate)
        print ("Recoveryrate:", self.recrate)
        print ("Population: ", self.population)
        print ("Recovery Time:", self.rect)
        print ("Uninfected People:", len (self.clean))

    def simulate(self, nDays = np.inf, initInfected=5, randomInfection = True):
        day = 0
        self.totInfect += initInfected
        newintections = r.sample(self.clean, initInfected)
        for p in newintections:
            self.clean.remove(p)
            self.infected.append(p)

        while day < nDays and (len(self.clean) > 0 or len(self.infected) > 0):
            day += 1
            self.infectCt.append(self.totInfect)
            self.deathCt.append(self.totDead)
            self.recCt.append(self.totRec)

          # Simulating Death
            randFact = (r.random()*(self.deathrate/3)-self.deathrate/1.5)*len(self.infected)
            deaths = min(round(len(self.infected) *self.deathrate+randFact), len(self.infected))
            if deaths == 0 and len(self.infected) > 0:
                for i in range(len(self.infected)):
                    if r.random() < self.deathrate: deaths += 1
            self.population      -= deaths
            self.totDead  += deaths
            newDeaths     = r.sample(self.infected, deaths)
            for p in newDeaths : self.infected.remove(p)

          # Simulating recovery
            randFact = (r.random()*(self.recrate/16)-self.recrate/8)*len(self.infected)
            recover  = min(round(len(self.infected)*self.recrate + randFact), len(self.infected))
            newRec   = r.sample(self.infected, recover)
            for p in self.infected:
                if p.daysSick >= self.rect:
                    if p not in newRec: newRec.append(p)
                else: p.daysSick += 1
            for p in newRec:
                p.recovered = True
                self.infected.remove(p)
            self.totRec += len(newRec)

          # Simulating Infection
            randFact = (r.random()*(self.infrate/3)-self.infrate/1.5)*len(self.infected)
            nNewInfect = min(round(len(self.infected)*self.infrate+randFact), len(self.clean))
            if nNewInfect == 0:
                if len(self.clean)>0 and randomInfection: nNewInfect = 1
                elif len(self.infected) == 0: break
            newInfects = r.sample (self.clean, nNewInfect)
            for p in newInfects:
                self.clean.remove(p)
                self.infected.append(p)
            self.totInfect += nNewInfect

            self.deathrate *= r.random()*.005+1
            self.infrate *= r.random()*.2+1
            self.recrate *= r.random()*.015+1

        self.infectCt.append(self.totInfect)
        self.deathCt.append(self.totDead)
        self.recCt.append(self.totRec)

        for x in range(len(self.infectCt)):
            self.cmrate.append(self.deathCt[x]/self.infectCt[x]*100)
            self.crrate.append(self.recCt[x]/self.infectCt[x]*100)
            if x >= self.rect:
                self.pmrate.append(self.deathCt[x]/self.infectCt[x-self.rect+1]*100)
```

```
                self.prrate.append(self.recCt[x]/self.infectCt[x-self.rect+1]*100)
            else:
                self.pmrate.append(0)
                self.prrate.append(0)

    def plot(self):
        pmratePlt = np.ma.masked_where(np.array(self.pmrate)==0,np.array(self.pmrate))
        cmratePlt = np.ma.masked_where(np.array(self.cmrate)==0,np.array(self.cmrate))
        prratePlt = np.ma.masked_where(np.array(self.prrate)==0,np.array(self.prrate))
        crratePlt = np.ma.masked_where(np.array(self.crrate)==0,np.array(self.crrate))
        arr = np.add(np.array(self.prrate), np.array(self.pmrate))
        addPlt = np.ma.masked_where(arr==0,arr)

        plt.plot(pmratePlt, label="Progressive Mortality Rate",color="r")
        plt.plot(cmratePlt, label="Mortality Rate",color="b")
        plt.xlim([0,45])
        plt.legend()
        plt.title("Mortality Rate and Progressive Mortality Rate")
        plt.xlabel("Days")
        plt.ylabel("Percentage")
        plt.show()

        plt.plot(prratePlt, label="Progressive Recovery Rate",color="g")
        plt.plot(crratePlt, label="Recovery Rate",color="purple")
        plt.xlim([0,45])
        plt.legend()
        plt.title("Recovery Rate and Progressive Recovery Rate")
        plt.xlabel("Days")
        plt.ylabel("Percentage")
        plt.show()


```

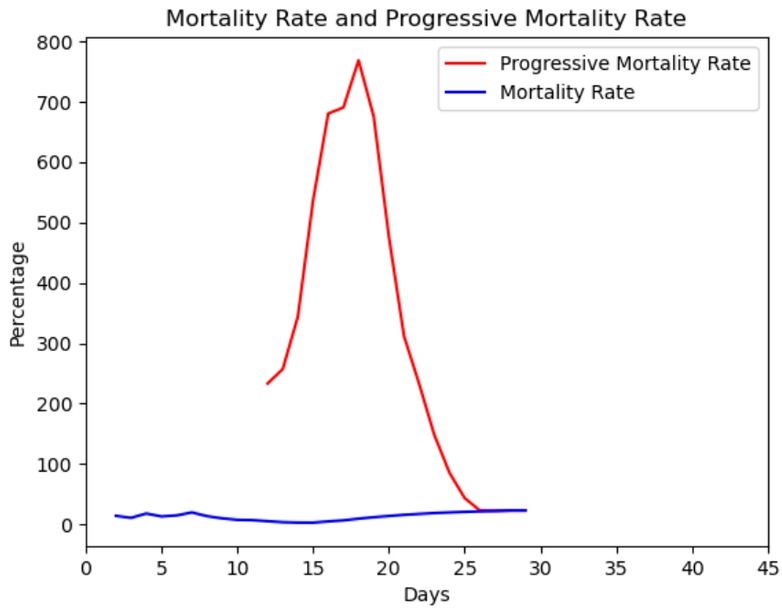
In [3]:

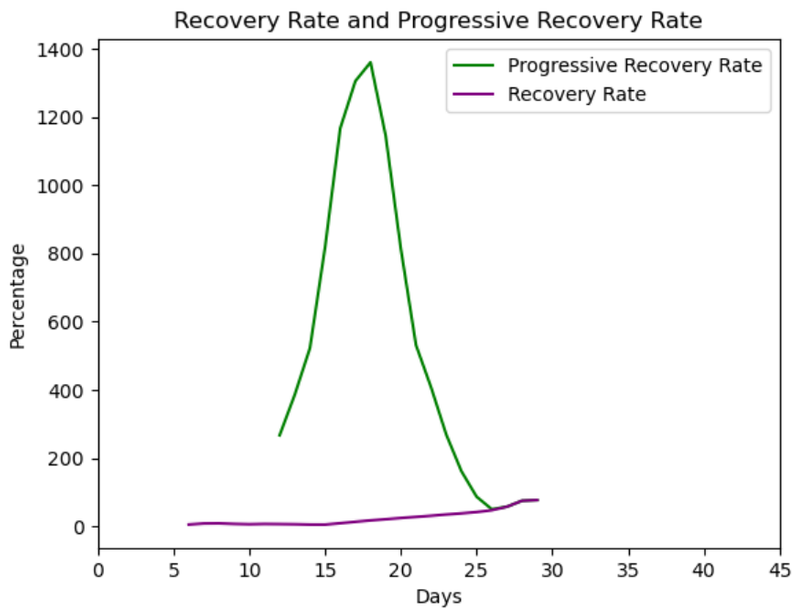
```
init_deathrate = .05
infect_rate    = .5
rec_rate       = .05

recTime = 12
population = 2000
diseaseX = Outbreak(recTime, population, init_deathrate, infect_rate, rec_rate, "diseaseX")
diseaseX.info()


```

```
Name: diseaseX
Deathrate: 0.05
Infectrate: 0.5
Recoveryrate: 0.05
Population:  2000
Recovery Time: 12
Uninfected People: 2000
```

In [4]:

```
1 diseaseX.simulate()
2 diseaseX.plot()
3 diseaseX.info()
```






```
Name: diseaseX
Deathrate: 0.052966971793434986
Infectrate: 10.526719486081126
Recoveryrate: 0.06098957711128803
Population:  1536
Recovery Time: 12
Uninfected People: 0
```

In [ ]:

```
1
```