> Math/Stat 2300 Assignment #2
Question 1: Logistic Map

```
> restart;
```

> (a) Equilibria:

```
> solve(x=r*x*(1-x),x);
```

$$0,\ \frac{-1+r}{r} \tag{1}$$

> The logistic map has equilibria: x=0 and x=(r-1)/r.

>

> (b) Write a procedure that accepts r as in input and returns ordered the ordered pairs (r; x181); (r; x182); : : : ; (r; x200). Use x1 = 0:1.

```
> LogisticMap:=proc(r)
   x[1]:=0.1;
   for i from 2 to 200 do
   x[i]:=r*x[i-1]*(1-x[i-1]); # Calculating the values of the sequence a_k
   end do;
   seq([r,x[j]],j=181..200); # Returning the last 20 values
   end proc:
Warning, `x` is implicitly declared local to procedure
`LogisticMap`
Warning, `i` is implicitly declared local to procedure
`LogisticMap`
```

>

> (c) Try various values of r varying between r = 2 and r = 4. Show the output for four different values in this range. Why would we choose to output the last 20 values of the sequence (instead of outputting just the last value of the sequence)?

```
> LogisticMap(2);
```

[2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000], [2, 0.5000000000] **(2)**

```
> LogisticMap(2.4);
```

[2.4, 0.5833333332], [2.4, 0.5833333335], [2.4, 0.5833333331], [2.4, 0.5833333332], [2.4, 0.5833333335], [2.4, 0.5833333331], [2.4, 0.5833333332], [2.4, 0.5833333335], [2.4, 0.5833333331], [2.4, 0.5833333332], [2.4, 0.5833333335], [2.4, 0.5833333331], [2.4, 0.5833333332], [2.4, 0.5833333335], [2.4, 0.5833333331], [2.4, 0.5833333332], [2.4, 0.5833333335], [2.4, 0.5833333331], [2.4, 0.5833333332], [2.4, 0.5833333335] **(3)**

```
> LogisticMap(2.6);
```

[2.6, 0.6153846153], [2.6, 0.6153846155], [2.6, 0.6153846152], [2.6, 0.6153846157], [2.6, 0.6153846153], [2.6, 0.6153846155], [2.6, 0.6153846152], [2.6, 0.6153846157], [2.6, 0.6153846153], [2.6, 0.6153846155], [2.6, 0.6153846152], [2.6, 0.6153846157], [2.6, 0.6153846153], [2.6, 0.6153846155], [2.6, 0.6153846152], [2.6, 0.6153846157], [2.6, **(4)**

[2.6, 0.6153846153], [2.6, 0.6153846155], [2.6, 0.6153846152], [2.6, 0.6153846157]

```
> LogisticMap(3.0);
```

[3.0, 0.6493105563], [3.0, 0.6831190734], [3.0, 0.6494022148], [3.0, 0.6830369345], [3.0, 0.6494924420], [3.0, 0.6829560294], [3.0, 0.6495812739], [3.0, 0.6828763276], [3.0, 0.6496687465], [3.0, 0.6827977991], [3.0, 0.6497548938], [3.0, 0.6827204152], [3.0, 0.6498397497], [3.0, 0.6826441482], [3.0, 0.6499233455], [3.0, 0.6825689712], [3.0, 0.6500057124], [3.0, 0.6824948587], [3.0, 0.6500868796], [3.0, 0.6824217858] **(5)**

```
> LogisticMap(3.4);
```

[3.4, 0.8421543989], [3.4, 0.4519632488], [3.4, 0.8421543999], [3.4, 0.4519632466], [3.4, 0.8421543989], [3.4, 0.4519632488], [3.4, 0.8421543999], [3.4, 0.4519632466], [3.4, 0.8421543989], [3.4, 0.4519632488], [3.4, 0.8421543999], [3.4, 0.4519632466], [3.4, 0.8421543989], [3.4, 0.4519632488], [3.4, 0.8421543999], [3.4, 0.4519632466], [3.4, 0.8421543989], [3.4, 0.4519632488], [3.4, 0.8421543999], [3.4, 0.4519632466] **(6)**

```
> LogisticMap(3.5);
```

[3.5, 0.8269407063], [3.5, 0.5008842110], [3.5, 0.8749972633], [3.5, 0.3828196839], [3.5, 0.8269407075], [3.5, 0.5008842082], [3.5, 0.8749972638], [3.5, 0.3828196825], [3.5, 0.8269407063], [3.5, 0.5008842110], [3.5, 0.8749972633], [3.5, 0.3828196839], [3.5, 0.8269407075], [3.5, 0.5008842082], [3.5, 0.8749972638], [3.5, 0.3828196825], [3.5, 0.8269407063], [3.5, 0.5008842110], [3.5, 0.8749972633], [3.5, 0.3828196839] **(7)**

```
> LogisticMap(4.0);
```

[4.0, 0.06297418520], [4.0, 0.2360337488], [4.0, 0.7212872729], [4.0, 0.8041277715], [4.0, 0.6300251944], [4.0, 0.9323737954], [4.0, 0.2522116042], [4.0, 0.7544036438], [4.0, 0.7411151440], [4.0, 0.7674539493], [4.0, 0.7138735400], [4.0, 0.8170324356], [4.0, 0.5979617390], [4.0, 0.9616139908], [4.0, 0.1476500940], [4.0, 0.5033981750], [4.0, 0.9999538096], [4.0, 0.0001847530658], [4.0, 0.0007388757284], [4.0, 0.002953319165] **(8)**

> For different values of r the sequence alternates between different values of x. We output the last 20 values to demonstrate the values of the sequence.

>

> (d) For the value r=2.5, the sequence x_n appraoches an equilibrium value.

```
> LogisticMap(2.5);
```

[2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000], [2.5, 0.6000000000] **(9)**

> This equlibrium value is x=0.6. Does this agree with part (a)?

```
> x_equilibrium:=(2.5-1)/2.5;
```

$$x\_equilibrium := 0.6000000000 \quad \textbf{(10)}$$

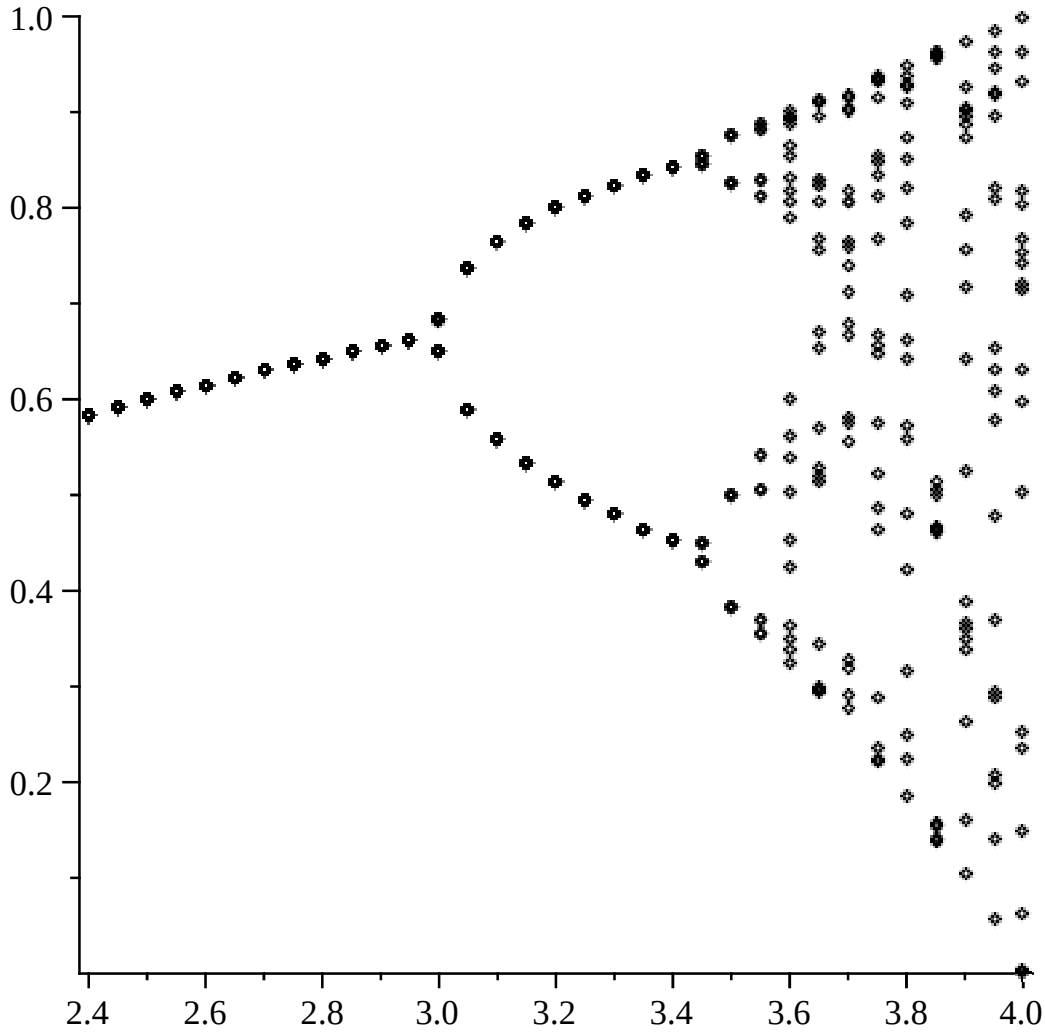
> This matches.

>

> (e)Plot the ordered pairs between 2.4 and 4.0 in steps of 0.05.

>

```
> with(plots):
> pointplot([seq(LogisticMap(r),r=2.4..4,0.05)]);
```

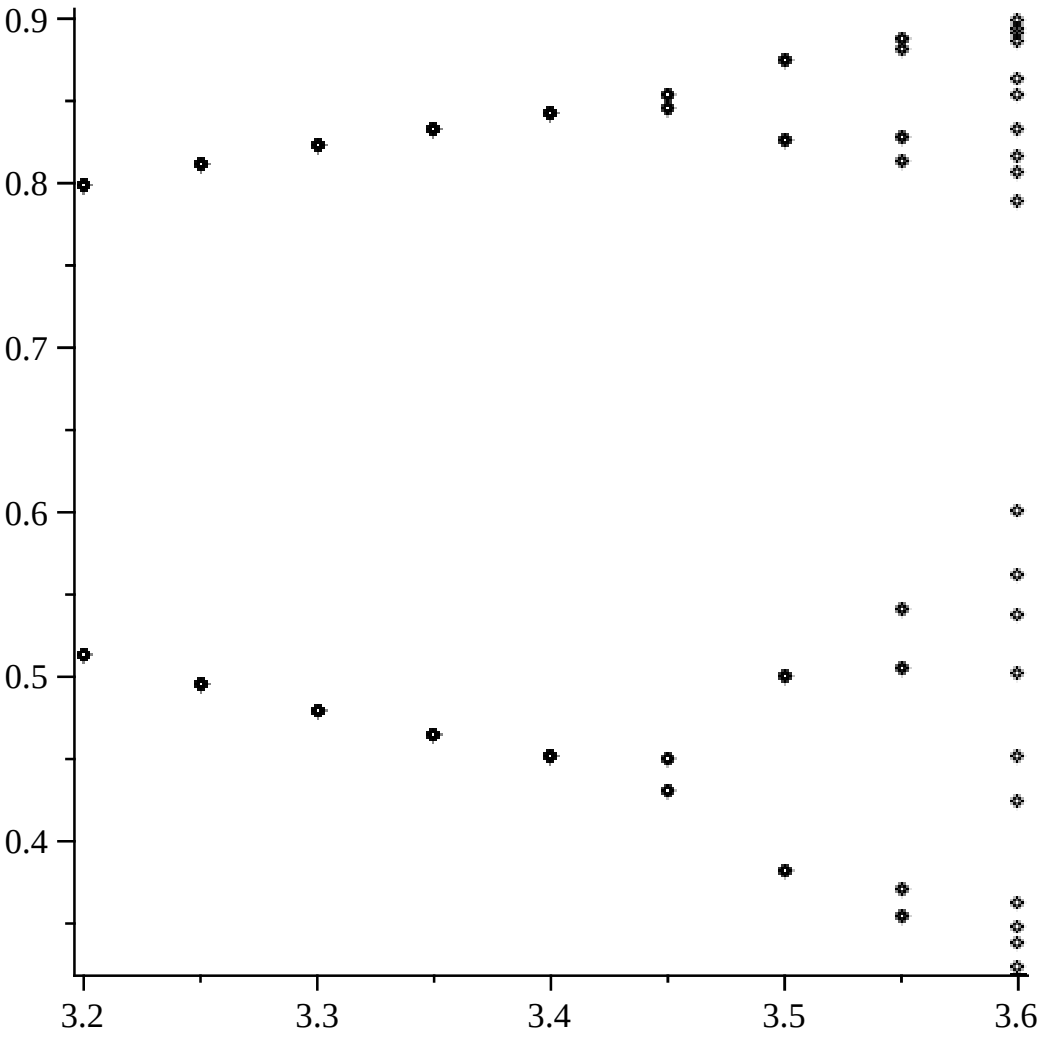


> The sequence seems to alternate between two values when r is between 3.0 and 3.45

>

> (f) Focusing on the region around r=3.45:

```
> pointplot([seq(LogisticMap(r),r=3.2..3.6,0.05)]);
```



> Near the value r=3.45, the sequence seems to change from alternating between 2 values to alternating between 4 values.

>

>

>

>