This problem is about predicting the world population.
Form the matrix of data for the world population from http://en.wikipedia.org/wiki/World_population (http://en.wikipedia.org/wiki/World_population), table "Estimated world and regional populations at various dates (in millions)" by taking the first two columns from 1950 until 2010.

In [1]:
```
A=[1950 2519
   1955 2756
   1960 2982
   1965 3335
   1970 3692
   1975 4068
   1980 4435
   1985 4831
   1990 5263
   1995 5674
   2000 6070
   2005 6454
   2010 6972];
```

Column 1 of A is time t and column 2, population P.

In [2]:
```
t=A[:,1]; P=A[:,2];
```

In [3]:
```
function B(k)
 z=zeros(Int,13,k+1)
 for i=0:k z[:,i+1]=t.^i end
 z
end
```

Out[3]:
```
B (generic function with 1 method)
```

Now approximate the population in the least-squares sense with the function $a\exp(*b*t)$, and with polynomials of second (quadratic), third (cubic) and fourth (quartic) degree. Plot all aproximations and the original data. Predict the current population by each approximation and compare to the current population from http://www.worldometers.info/world-population/ (http://www.worldometers.info/world-population/) (Looks like 7266 so far this year on October 10, 2014)

Suppose we say 7266 + 63*(80/365) is the actual number this year -- you might be able to be a bit more accurate.

In [40]:
```
actual = 7266 + 63*(80/365)
```

Out[40]:
```
7279.808219178082
```

In [4]:
```
B(2)  # Quadratic
```

Out[4]:
```
13x3 Array{Int64,2}:
 1  1950  3802500
 1  1955  3822025
 1  1960  3841600
 1  1965  3861225
 1  1970  3880900
 1  1975  3900625
 1  1980  3920400
 1  1985  3940225
 1  1990  3960100
 1  1995  3980025
 1  2000  4000000
 1  2005  4020025
 1  2010  4040100
```

In [5]:
```
c=B(2)\P
```

Out[5]:
```
3-element Array{Float64,1}:
  -0.0714
 -70.6796
   0.0368522
```

Evaluate the polynomial and plot it

In [6]:
```
z(k)=[B(k) ;2014.^(0:k)']
```

Out[6]:
```
z (generic function with 1 method)
```

In [7]:
```
z(2)
```

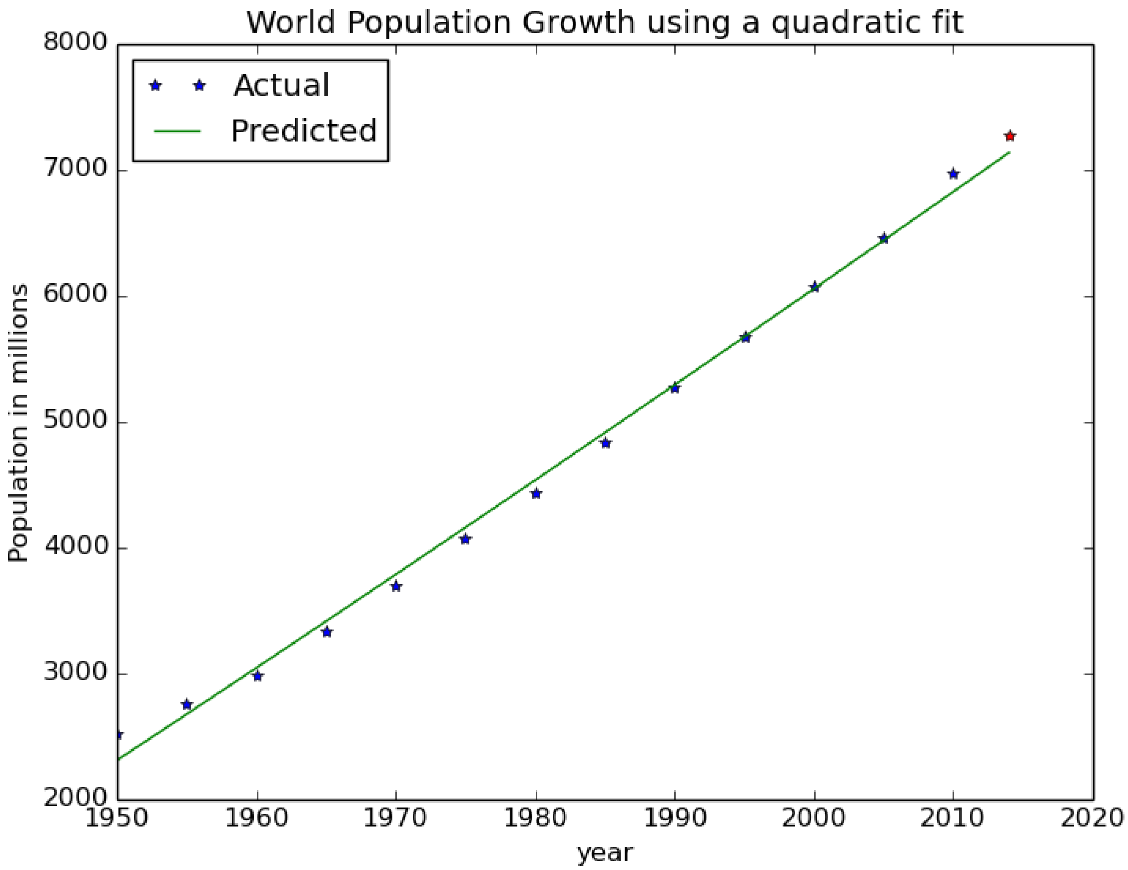
Out[7]:
```
14x3 Array{Int64,2}:
 1  1950  3802500
 1  1955  3822025
 1  1960  3841600
 1  1965  3861225
 1  1970  3880900
 1  1975  3900625
 1  1980  3920400
 1  1985  3940225
 1  1990  3960100
 1  1995  3980025
 1  2000  4000000
 1  2005  4020025
 1  2010  4040100
 1  2014  4056196
```

In [8]:
```
using PyPlot
```

```
INFO: Loading help data...
```

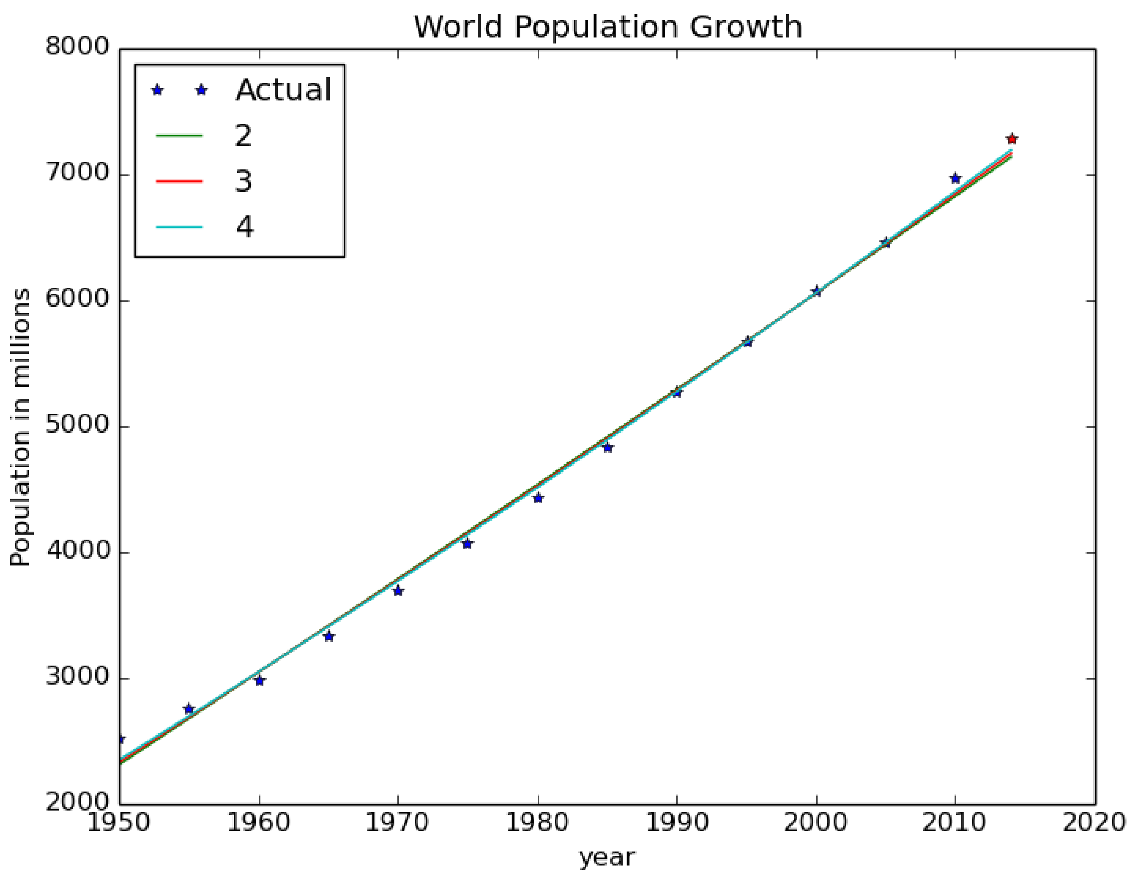
In [12]:

```
plot(t,P,"*")
plot([t;2014],z(2)*c)
plot(2014,7266,"r*")
xlabel("year")
ylabel("Population in millions")
title("World Population Growth using a quadratic fit")
legend({"Actual","Predicted"},loc="upper left")
savefig("PopulationGrowth")
```



In [43]:
```
nmax=4
plot(t,P,"*")
for n=2:nmax
    c=B(n)\P
    plot([t;2014],z(n)*c)
    println("degree $(n): 2014 Estimate: $(round(z(n)*c)[end])
 Actual: $(round(actual))")
end
plot(2014,actual,"r*")
xlabel("year")
ylabel("Population in millions")
title("World Population Growth")
legend({"Actual", (2:nmax)...},loc="upper left")
#savefig("PopulationGrowth")
```

```
degree 2: 2014 Estimate: 7131.0  Actual: 7280.0
degree 3: 2014 Estimate: 7160.0  Actual: 7280.0
degree 4: 2014 Estimate: 7188.0  Actual: 7280.0
```



Out[43]: PyObject <matplotlib.legend.Legend object at 0x7f1e47d6cc50>

Do polynomials keep getting better? Or eventually do they breakdown? Submit graphs and writeup as evidence. In juliabox you can run savefig("PopulationGrowth") and then you can see the file by going file-->open in ijulia or by clicking on the home page. The browser print button will let you print

In [ ]:
```
We were hoping for better luck with an exponential fit.  It was
 not a good fit.

Nonetheless explain how this works:
```

In [44]:
```
c=[t.^0 t]\log(P)
```

Out[44]:
```
2-element Array{Float64,1}:
 -25.7489
   0.017232
```

In [45]:
```
exp(c[1] + 2014 * c[2] )
```

Out[45]:
```
7756.98309596893
```

In [ ]: