CODE FRAGMENT 1

```
library(haven)
library(dplyr)
library(purrr)
library(magrittr)

#set the working directory.
setwd()

#create two data frames: whites and hispanics
la_data_white <- read_dta('la_data_set.dta') %>%
 #clean out bad observations
 filter(!is.na(rent_90)) %>%
 filter(nwnh00!=0 & nwnh10!=0 & nh00!=0 & nh10!=0) %>%
 #here is where you pick the sample size
 head(20) %>%
 #moving cost of leaving the neighborhood
 mutate(mc = (2910+(1-rent_05/100)*home_value05*.03)/24.556) %>%
 dplyr::select(FIPS, nwnh00, nwnh10, mc) %>%
 #create the outside option
 rbind(as_tibble(list(FIPS = 1,
                      nwnh00 = 4*abs(sum(.$nwnh00)-sum(.$nwnh10)),
                      nwnh10 = 4*abs(sum(.$nwnh00)-sum(.$nwnh10))-
                               (sum(.$nwnh10)-sum(.$nwnh00)),
                      mc = 528.71))) %>%
 rename(n00 = nwnh00, n10=nwnh10) %>%

 #here is where I make sigmas from equation 11

 mutate(sigma_00 = #fill this in ,
        sigma_10 = #fill this in )
```

CODE FRAGMENT 2

```
deltas <- function(df, delta_guess, mu_guess, counter = 1, kill = 1000){

 #add the new delta_guess
 df_original <- df

 df <- df %>%
  mutate(delta_0 = delta_guess)

 #Create the denominator for equation 11
 denom <- vector("double", nrow(df))

 for (l in 1:nrow(df)) {

** denom for eqn 11

   denom[[l]] <- map2_dbl(df$delta_0, df$mc,
                    ~ if(.x==df$delta_0[[l]]){
                     #fill this in
                    }else{
                     #fill this in
                    }
  ) %>%
    sum()
 }

 #Create the bigger summation for equation 11
 sigma_10_bar <- vector("double", nrow(df))

 for (k in 1:nrow(df)) {
  sigma_10_bar[[k]] <- pmap_dbl(list(df$delta_0, df$mc, denom, df$sigma_00),
                          ~ if(df$delta_0[[k]]==..1){
                           #fill this in
                          }else{
                           #fill this in
                          }
  ) %>%
    sum()
 }

#create the new guess
 df <- df %>%
```

```
   mutate(denom = denom,
          sigma_10_bar = sigma_10_bar,
          delta_1 = #fill this in,
          stay = 1/denom,
          delta_fail = abs(delta_1 - delta_0) > 10e-8)

  #either return the final estimates, or continue the recursive function
  if(sum(df$delta_fail)==0 | counter== kill){
    print(counter)
    return(df)
  }else{
    print(counter)
    counter <- counter + 1
    deltas(df_original, df$delta_1, mu_guess, counter, kill = kill)
  }
}
```

To run the function, use this code:

```
set.seed(08241987)
delta_df <- deltas(la_data_white, rnorm(nrow(la_data_white),1000,10), mu_guess=.003,
                   kill=1000)
```

CODE FRAGMENT 3 (OPTIONAL)

```
mu_loop <- function(df, delta_guess, mu_upper, mu_lower,
                counter_mu=1, delta_kill = 1000, mu_kill = 50, stay, last_guess = 0){
  mu_guess <- #fill in yourself
  df_deltas <- deltas(df, delta_guess, mu_guess = mu_guess, kill = delta_kill)
#equation 10
 pred_stay <- df_deltas %$%
    #fill in yourself
  guess_diff <- pred_stay - stay
  print(paste(counter_mu,' mu=',mu_guess,' and diff=',guess_diff,sep=''))
  mu_diff= mu_guess-last_guess

  if(abs(mu_diff)<10e-7 | counter_mu== mu_kill){
    return(df_deltas)
  }else if(guess_diff>0){
    counter_mu <- counter_mu + 1
    set.seed(08241987)
    mu_loop(df, delta_guess, mu_guess, mu_lower,
          counter_mu, delta_kill, mu_kill, stay, last_guess = mu_guess)
  }else{
    counter_mu <- counter_mu + 1
    set.seed(08241987)
    mu_loop(df, delta_guess, mu_upper, mu_guess,
          counter_mu, delta_kill, mu_kill, stay, last_guess = mu_guess)
  }
}
```

***NOTE**

Once you have completed the function, you can run it with these equations:

```
white_2loops <- mu_loop(la_data_white, rnorm(nrow(la_data_white),1000,10),
                mu_upper = .005, mu_lower = .001, mu_kill = 50, stay = 0.4506)

hisp_2loops <- mu_loop(la_data_hisp, rnorm(nrow(la_data_hisp),1000,10),
                   mu_upper = .005, mu_lower = .001, mu_kill = 50, stay = 0.393)
```