In the following parts, we will give the codes we used to finish our assignment; As we have a macro, we think it is easy to produce results for other cells, so we add two new parameters in our macro. But at the beginning, we take the first cycle as an example to write the basic codes.

```
libname worklib '/folders/myfolders/sasuser.v94/l5';
data param;
      year_cycle=1965;
      num=1;
run;
proc sql;
create table msf_data as
select
      msf.permno,
      msf.ret,
       msf.date,
      msf.hexcd,
       param.*
from
       param left join worklib.msf
on
      (hexcd=1 or hexcd=2) and date>=mdy(1,1,year_cycle-1) and date<mdy(1,1,year_cycle+1);
quit;
```

Step1: We should get the data we need for all NYSE and Amex securities for months in the window[T-12, T+12], but we didn't download it using the codes in our email-box, we have tried and we failed. we just use the dataset "MSF" in our email-box. "Num" actually is a variable we used to make our code flexible to month, for we have to cycle quarterly. We will see it again in our macro.

```
data msf_data;
       set msf_data;
       month=year(date)*100+month(date);
       keep year_cycle permno month ret;
run;

proc print data=msf_data(obs=5);
      title 'Generating Data for This Year';
      var year_cycle month;
run;
data d;
      set msf_data;
run;

* Number months;

proc sort data=d out=d_months(keep=month) nodupkeys;
       by month;
run;

data d_months;
       set d_months;
       i+1;
run;
```

Step2: In this step, we check whether the time window we chosen is right or not, and generate a new variable month to describe month.

```
* Get history average return;
proc sort data=d;
      by month;
run;

data d;
      merge d d_months;
      by month;
run;

proc sort data=d;
      by permno;
run;
proc means data=d noprint;
      var ret;
      output out=d_stats n=n_ret;
      by permno;
      where i>4 and i<=16;
run;
* Keep only securities with returns in all 12 months;
data d;
       merge d d_stats;
       by permno;
       if n_ret=12;
run;
```

Step3: This restriction is imposed to avoid small thinly traded stocks, from contaminating the results of the experiment. The code is simple: t is the 12th month in the time window(according to our selection method), so the data we need can be defined as i>4 and i<=16, and we use "proc means", it counts the number of data points for monthly data in this period, so we keep in the sample the permnos with 12 months of valid data.

```
*Get history abnormal return;
proc sort data=d;
      by year_cycle permno;
run;
proc means data=d noprint;
      output out=d_avgret
      mean(ret)=avgre;
      n(ret)=n_avgret;
      where i>4 and i<=13;
      by year_cycle permno;
run;

proc sort data=d_avgret;
      by avgret;
run;
```

Step4: compute monthly average return amongst all securities

```
proc sort data=d_avgret;
      by avgret;
run;
* Define winners and losers;
data winner loser;
      set d_avgret nobs=num_records;
     percentile=_n_/num_records;
      if percentile<=0.1 then output loser;
      if percentile>=0.9 then output winner;
run;
```

Step 5: Compute for each stock the average raw returns for the 9 months preceding T. Choose winners (the top 10% of stocks based on the average raw return defined above) and losers (the bottom 10%).

```
proc sort data=winner;
      by permno;
run;
proc sort data=loser;
           by permno;
run;
data d;
       merge
          d(keep=permno i month ret year_cycle)
          winner(keep=permno in=in_winner)
          loser(keep=permno in=in_loser);;
       by permno;
       winner=in_winner;
       if in_winner or in_loser;
       if i>13 and i<=16;
run;
```

Step 6: For the months T+1, T+2 and T+3, keep the stocks of winners and losers in the same dataset throughout all cycles.

```
proc means data=d noprint;
        var ret;
        output out=winner_stats
            mean(ret)=avg_ret_winner
            n(ret)=n_avg_ret_winner;
        by year_cycle month;
        where winner=1;
run;

proc means data=d noprint;
        var ret;
        output out=loser_stats
            mean(ret)=avg_ret_loser
            n(ret)=n_avg_ret_loser;
        by year_cycle month;
        where winner=0;
run;

data d_stats;
        merge loser_stats winner_stats;
        by year_cycle month;
run;
```

Step 7: Compute for each month the average return on the two portfolios, then compute the average of these monthly returns on the two portfolios. These are the numbers that should mimic the "Buy" and "Sell" results in the table.

Now, let's turn to our macro codes; Since what we have done is the same thing as the first cycle, we just show the different parts;

```
%macro run_cycle(first_year,num_cycles,j,k);
%do i=1 %to &num_cycles;
      %let num1=mod(&i-1,4);
      %let num2=(&i-1-&num1)/4;
      %let num=&num1*3+1;
      %let year_cycle=&first_year+&num2+1;
```

See, we meet what we talk about above again. We use "%j" and "%k" to make our code more flexible to accommodate the other cells in the table. And we generate macro variable "num1","num2",and "num" to control the year_cycle and to realize cycle quarterly. As we use I to control the cycle times, we can use I mod 4, then we can get number 0, 1,2 ,3 and we just do some simple calculation to ensure that we can get month 1,4,7,10.

```
*How to use &num and &year_cycle.
data param;
      year_cycle=&year_cycle;
      num=&num;
run;

proc sql;
create table msf_data as
select
      msf.permno,
      msf.ret,
       msf.date,
      msf.hexcd,
       param.*
from
       param left join worklib.msf
on
       (hexcd=1 or hexcd=2) and date>=mdy(num,1,year_cycle-1) and
date<mdy(num,1,year_cycle+1);
quit;
```

we define the time window as [T-12,T+12], so time t's order is 13, so we use 13-&j and 13+&k to make it flexible to the other cells. And we then append all the results in dataset: d_cycles.

```
%if &i=1 %then
   %do;
         data d_cycles;
              set d_stats;
```

```
            run;
        %end;
        %else
        %do;
           proc append base=d_cycles data=d_stats;
           run;
        %end;
```

```
proc means data=d noprint;
      var ret;
      output out=d_stats n=n_ret;
      by permno;
      where i>13-&j and i<=13+&k;
run;
* Keep only securities with returns in all 12 months;
data d;
       merge d d_stats;
       by permno;
       if n_ret=&j+&k;
run;

* Compute past average returns;

proc sort data=d;
       by year_cycle permno;
run;

proc means data=d noprint;
       output out=d_avgret
          mean(ret)=avgret
          n(ret)=n_avgret;
       where i>13-&j and i<=13;
       by year_cycle permno;
run;
```

The results we get are shown below:

观测	_TYPE_	_FREQ_	avg_ret_winner	avg_ret_loser	ret_buy_sell
1	0	300	0.019337	0.009798	.009539864

They are slightly different from the ones in the table.

We repeat what Jegadeesh and Titman (1993) done, and the results show that, at least in some situations, winners keep winning and losers keep losing. This came to be known as the momentum effect. We will analyze in this assignment the momentum. But we also have seen in class that how supposedly winners become losers and vice versa, we guess the difference is that how time window is defined as and test period is chosen, but we have no evidence, this is left for further investigation.