```
#include <stdio.h>
#include <iostream.h>
#define mhigh 20
#define rowhigh 20
#define colhigh 20

int m;
extern int m;
long double balance[mhigh],min[mhigh];
extern long double balance[],min[];
float rate[mhigh];
extern float rate[];
long double pay;
extern long double pay;
int n;
extern int n;
int horimax,vertimax;
extern int horimax,vertimax;
long double x[rowhigh][colhigh];
int xrow[rowhigh],xcol[colhigh];
extern int xrow[],xcol[];
long double runit[30][30];
long double xvalue[rowhigh];
extern long double xvalue[];
int xrowpoint[rowhigh];


extern int xrowpoint[];

//extern long double x[][];

void userinput(void);
void powergen(void);
void loadarray(void);
int pivotcolfind(void);
void linpointsort(void);
int pivotrowfind(int pivotcol);
void pivotoperations(int pivotrow,int pivotcol);


////////////////////////////////////////////////////////////////
void main(void)
{int looprow,loopcol;
 char button;
 userinput();
 n++;
 horimax=1+m*(n-2);
 vertimax=1+m+n-1;
 for (looprow=1;looprow<=m;looprow++)
 {cout << "Rate " << looprow << " = " << rate[looprow] << endl;
  cout << "Balance " << looprow << " = " << balance[looprow] << endl;
  cout << "Min " << looprow << " = " << min[looprow] << endl;
 }
 cout << "Pay = " << pay << endl;
 cout << "M = " << m << endl;
 cout << "N (really n+1) = " << n << endl;
 cout << "Horimax = " << horimax << endl;
 cout << "Vertimax = " << vertimax << endl;
 loadarray();
```

```
for (loopcol=1;loopcol<=horimax;loopcol++) //print names
  cout << xcol[loopcol] << " ";
cout << endl;

for (looprow=1;looprow<=vertimax;looprow++)
{cout << xrow[looprow] << " "; // print names
```

```
 for (loopcol=1;loopcol<=horimax;loopcol++)
 {cout << x[looprow][loopcol] << " ";

  if (loopcol==horimax)
    cout << endl;
 }
}
int pivotrow,pivotcol;

pivotcol=pivotcolfind();
while (x[1][pivotcol] > 0)
{pivotrow=pivotrowfind(pivotcol);
 pivotoperations(pivotrow,pivotcol);
 pivotcol=pivotcolfind();
}

//////print
for (loopcol=1;loopcol<=horimax;loopcol++) //print names
  cout << xcol[loopcol] << " ";
cout << endl;

for (looprow=1;looprow<=vertimax;looprow++)
{cout << xrow[looprow] << " "; // print names

 for (loopcol=1;loopcol<=horimax;loopcol++)
 {cout << x[looprow][loopcol] << " ";

  if (loopcol==horimax)
    cout << endl;
 }
}

cout << "Finished-hit a key";
cin >> button;


////////////////////////////////////////////////////////////////

void userinput(void)
{extern int m,n;
 extern long double balance[],min[],pay;
 extern float rate[];

 int loop;

 cout << "Enter number of cards (<= " << mhigh << "): ";
 cin >> m;
 for (loop=1;loop<=m;loop++)
 {  cout << "Enter balance of card " << loop << ": ";
    cin >> balance[loop];
    cout << "Enter interest rate of card" << loop << ": ";
    cin >> rate[loop];
    cout << "Enter minimum payment of card"  << loop << ": ";
    cin >> min[loop];
 }
   cout << "Enter payment amount: ";
  cin >> pay;
    cout << "Enter number of months to pay it off in: ";
```

```
  cin >> n;

////////////////////////////////////////////////////////////////////////
void loadarray(void)
```

```
extern int m,n;
extern long double x[][colhigh];
extern long double balance[],pay;
extern int xrow[],xcol[];

extern long double runit[][30];
int card,payment=0,count=1;

powergen();
x[m+2][1]=pay;
for (card=1;card<=m;count++)
{ ++payment;
  if (payment ==1)
  {x[1][1] -= runit[card][n]*balance[card];
   x[card+1][1]=balance[card];
   x[m+2][1] -= balance[card];
  }
  if (card==1)
    x[m+2+payment][1]=pay;
  x[1][count+1]=runit[card][2*n-1-payment]-runit[card][n-payment];
  x[card+1][count+1]=-(1/runit[card][payment]);
  x[m+2][count+1]=(1/runit[card][payment]);
  x[payment+m+2][count+1]=-1;
  xrow[count+1] = (count+1<=(1+m)) ? 1+(n-1)*(count+1-2) :-(count-m);
  xcol[count+1]=count+1+(card-1);
  if (payment==(n-2))
  {payment=0;
   ++card;
  }
}
return;

////////////////////////////////////////////////////////////////////
void powergen(void)
{extern int m,n;
 extern float rate[];
 extern long double runit[][30];

 int card, power;
 long double cache;
 float factor;

 for (card=1;card<=m;card++)
 {cache=1;
  factor=rate[card]+1;
  for (power=1;power<=2*n-2;power++)
    {cache*=factor;
     runit[card][power]=cache;
    }
  }
  return;
}

////////////////////////////////////////////////////////////////////

int pivotcolfind(void)
{extern long double x[][colhigh];
 extern int horimax;
```

is this working?

```
int column;
int maximumcol = 2;


for (column=2;column<=horimax;column++)
  if (x[1][column]>x[1][maximumcol])
```

-1

3.7 - 3.7

2.9 - 3.8

18 - 24 = -6

```
   return maximumcol;
}

////////////////////////////////////////////////////////////////
void linpointsort(void)
{extern long double xvalue[];
 extern int xrowpoint[];
 extern int vertimax;
 int row,nextelrow,nrow;
 for (row=2;row<=vertimax;row++)
 {nextelrow=row;
  xrowpoint[0]=nextelrow;
  nrow=row;
  while ((xvalue[nextelrow]>xvalue[xrowpoint[nrow-1]]))
  {xrowpoint[nrow]=xrowpoint[nrow-1];
   nrow--;
  }
  xrowpoint[nrow]=nextelrow;
 }

}

////////////////////////////////////////////////////////////////

int pivotrowfind(int pivotcol)
{extern long double x[][colhigh];
 extern int vertimax;
 extern long double xvalue[];
 extern int xrowpoint[];
 int row,top;
 int negative;
 int listsize;
 xvalue[0]=-9.9999999999999E307;   <- remove
 listsize=vertimax-1;
 for (row=1;row<=vertimax-1;row++)   xrowpoint[row]=row;
 {if (x[row+1][pivotcol]!=0)
  {xvalue[row]=-x[row+1][1]/x[row+1][pivotcol];
   xrowpoint[row]=row;
  }
  else {xrowpoint[row]=0;
        listsize--;
       } xvalue[row]=-9.999999E307

   cout << "xvalue " << row << ": " << xvalue[row] << " " << xrowpoint[row] <<

 }
 linpointsort();
 for (row=1;row<=listsize;row++)
   cout << "sorted new xvalue " << row << ": " << xvalue[xrowpoint[row]] << endl;   print only
 for (top=1;top<=vertimax-1;top++)   listsize <- loop for sorted ones to test
 {if (xrowpoint[top]==0)
    continue;
  negative=0;
  for (row=2;row<=vertimax;row++)   loop for rows to test sorted ones on
    if (x[row][1]+x[row][pivotcol]*xvalue[xrowpoint[top]]<0)
    {negative=1;
     break;
    }
  if (negative==0)
```

```
    break;
  }
  return xrowpoint[top+1];
}
```

since x values were stored from 1-(vert-1) and not 2-vert, must add 1

//

```
void pivotoperations(int pivotrow,int pivotcol)
{extern long double x[][colhigh];
 extern int horimax,vertimax;
 extern int xcol[];
 extern int xrow[];
 int row, column;
 long double pivotfactor;
 int temp;

 for (column=1;column<=horimax;column++)
   for (row=1;row<=vertimax;row++)
     if ((column != pivotcol) && (row != pivotrow))
       x[row][column]=x[row][column]*x[pivotrow][pivotcol]
        -x[pivotrow][column]*x[row][pivotcol];

   for (column=1;column<=horimax;column++)
     if (column != pivotcol)
       x[pivotrow][column]*=-1;

   pivotfactor=1/x[pivotrow][pivotcol];
   x[pivotrow][pivotcol]=1;

   if (pivotfactor != 1)
     for (column=1;column<=horimax;column++)
       for (row=1;row<=vertimax;row++)
         x[row][column]*=pivotfactor;

   temp=xcol[pivotcol];
   xcol[pivotcol]=xrow[pivotrow];
   xrow[pivotrow]=temp;
```