DOTS AND BOXES GAME

1 Game Description

Dots and Boxes is a game for two players (or more). It was first published in the 19th century
by Edouard Lucas, who called it originally la pipopipette. Starting with an empty grid of dots,
two players take turns adding a single horizontal or vertical line between two unjoined adjacent
dots. The player who completes the fourth side of a 1x1 box (or groups of one or more adjacent
boxes) earns one point (s) and takes another turn. In this project you will make your own
version of the game.

2 Data Structure

We used 6 arrays

- A to store dots, lines and boxes its size is 50*50.
- R to store colors of lines boxes which stored before .
- arr to store moves for redo that contain the two dots that were connected.
- playe to store the player who did that move to use it in undo & redo.
- type to store type of line vertical or horizontal to use in undo &redo.
- redo to store lines that were removed by undo to make it possible to redo it.

We used two structures

- player structure that contains
    - Name of player
    - Score of player
    - Moves of player

- Rank list structure that contains
    - Name of player
    - Score of player

3 User manual

When the game begun you should select
Start game, load game, rank, about game or exit by pressing the number of your choice.

- If you choose start game you should select the level of the game beginner of expert by pressing the number of your choice.
  2*2,3*3,4*4 for beginner
  5*5,6*6,7*7 for expert
  Then you should select vs computer or vs player
  Then you should enter your name and the game will start

  You can play by enter the number of your two dots in two digits like 01,02
  If you want to save the game you should enter save and choose the place you want to save in from 1,2 and 3 saved files

  If you want to load another saved file you should enter load

  and choose the saved file from 1,2 and 3.

  If you want to exit and return to menu you should press exit.

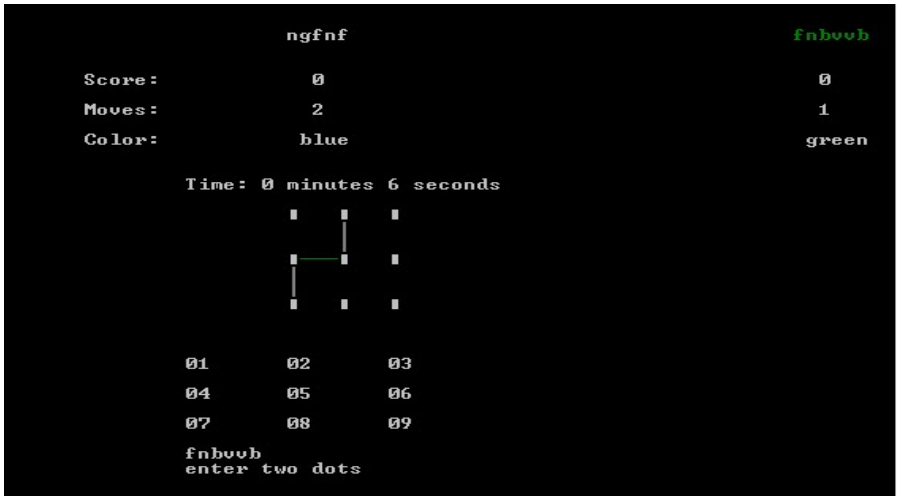
  After the game you should choose between return to menu

  or exit.

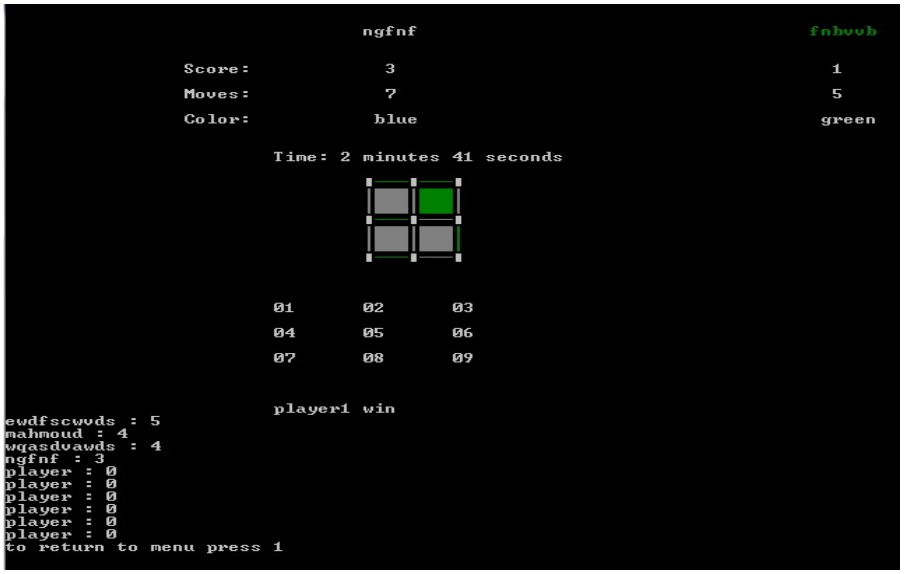
- If you choose load you should choose the saved file from 1,2 and 3.

4 Screenshots

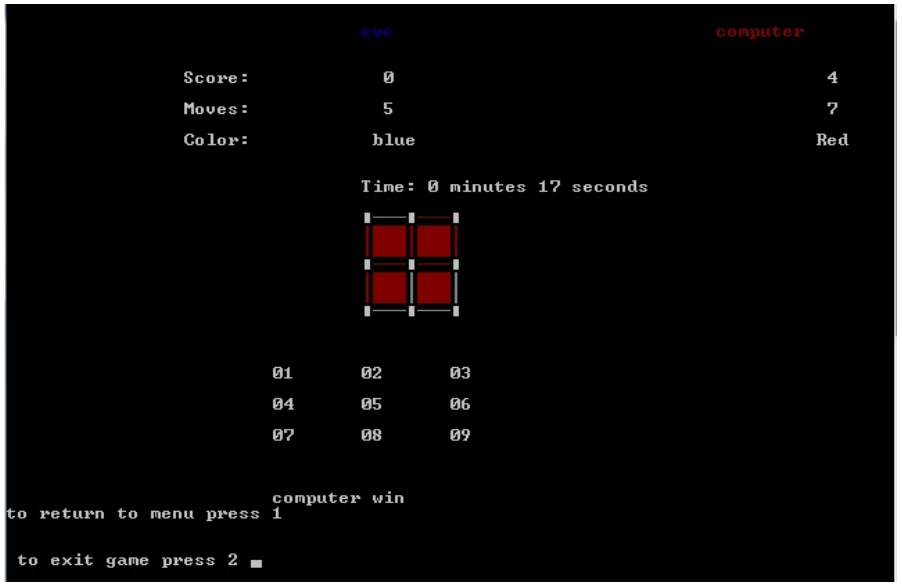
While playing



While winning



While losing



5-(Featuers of application)

•Game provides many levels to user to enable him to play game according to its level.

•Design of game shows to user how to play game easily.

•there are two modes in game .one is computer vs player and another is player vs player.

6-(Design of game)

•design contains main menu that consists of five options as following:

1. play game option that consists of the modes of game , levels and game implementation.

2. load game option that enable user to restore game that he saved before.

3. Rank option that shows top 10 players who play game before.

4. About game option that shows to user how to play game and information about game.

5. Exit option that enable user to leave game if he like

7-(Description of important functions)

Computer function

•this function make computer plays vs user with strategy as following:

It follows four steps respectively if one is implemented, he ignore others. Four orderd steps:

1. It check if there are boxes that can be completed or not .if it find ,it complete this box and if not ,it moves to next step.

2. it check if there are empty boxes . if it find any boxes without lines , it draw one line in it and if not , it moves to next step.

3. it check if there is one line in any box without other borders . if it find, it draw another line in this box and if not , it moves to next step.

4. it check if there are two lines in any box . if it find, it draw another line in this box .

By this strategy user plays against computer as if it is real player.

Play game function

- It’s role to organize flow of game whether mode is player vs player or computer vs player.

player1 function&&**player2 function**

•to enable one of two player to play game in his turn.

player vs player function

•To organize game if mode is player vs player . it determines turn of player.

player vs computer function

•To organize game if mode is player vs computer . It determines turn Of computer or player.

Boxes function

•To check if there is box completed or not for each line drawn by one of playres and update score of this player if any box is completed.

Rank function

•To register ranks of top 10 players.

8-(Assumptions)

•If user want to remove previous step he entered during playing (undo), he should enter undo.

• If user want to restore step he removed during playing (redo), he should enter redo.

•If user wants to save game that he play, he should enter save.

Pdeudcode of function player 1

-Declare I , j, a, q, b, n, m, min, score as integer

-Declare x ,y as character and num as string

-set q equel to 0.

-diplay “player_1str variable” and “enter two dots”

-scan num .

-check if num is “exit” word .if true ,put Exit that is global variable equal to 1 and return function.

-OR if num is “save” word. If true , implement following

-diplay "put the location of save from 1 , 2 and 3\n".
- Declare st as string .
-scan st.
-declare l as integer and set it to st[0]-‘0’.
-if l is less or equal to 3 OR greater than or equal to 1.
-call save function and pass l and play variables as arguments.

-call load function and regiser its return value in play variable.
-then call system ,print_board and digits functions ,than return this function.

-OR if num is “load” word. If true , implement following

-diplay "put the location of save from 1 , 2 and 3\n".
- Declare st as string .
-scan st.
-declare l as integer and set it to st[0]-‘0’.
-if l is less or equal to 3 OR greater than or equal to 1.
-set load_index = 1 .
-call load function and regiser its return value in play variable.
-then call system ,print_board and digits functions ,then return this function.

-OR if num is “undo” word. If true , implement following

- declare l as integer.
- call undo function and regiser its return value in play variable.
-then call system , print_board and digits functions ,then return this function.

-OR if num is “undo” word. If true , implement following

-put n=redo[redo_index-2], m=redo[redo_index-1],redo_index=redo_index-2.

-else put x=num[0],y=num[1],
-call valid function with x, y as arguments and restore its return value in n variable.
-if n=0 , return function.
-else if m!=0 ,scan num again .
-and else put x=num[0],y=num[1].
END IF.

-call valid function with x, y as arguments and restore its return value in m variable.

```
- if m=0 , return function.
-else
    for i=0 to 600 put redo[i]=0,
redo_index=0;
-if n is greater than m put min
equal to n.
-else  put min equal to m.

END IF.


-call system("cls") function.

-for each value of I from 0 to
value of row variable with update 3
in each step, implement following:
-for each value of j from 0 to
value of col variable with update 4
in each step, implement following:
-check if this condition:
(abs (m-
n)==1)&&(min!=(grid+1))&&(min!=(gri
d+1)*2)&&(min!=(grid+1)*3)&&(min!=(
grid+1)*4)&&(min!=(grid+1)*5)&&(min
!=(grid+1)*6)&&(min!=(grid+1)*7)&&(
min!=(grid+1)*8)&&(min!=(grid+1)*9)
```

and this condition
(B[i][j]==min&&A[i][j+3]!=196)

is true implement following:

```
q=1;

A[i][j+1]=A[i][j+2]=A[i][j+3]=196;

R[i][j+1]=R[i][j+2]=R[i][j+3]=1;

a=i;b=j;

arr[ind]=min;

type[ind]=1;

playe[ind]=0;

ind=ind+1;
```

-else if (abs(m-n)==(grid+1),if
true implement foolowing:

```
A[i+1][j]=A[i+2][j]=179;
```

```
R[i+1][j]=R[i+2][j]=1;

a=i;b=j;

arr[ind]=min;

type[ind]=2;

playe[ind]=0;

ind=ind+1;

END IF.
```

-then check if this condition:

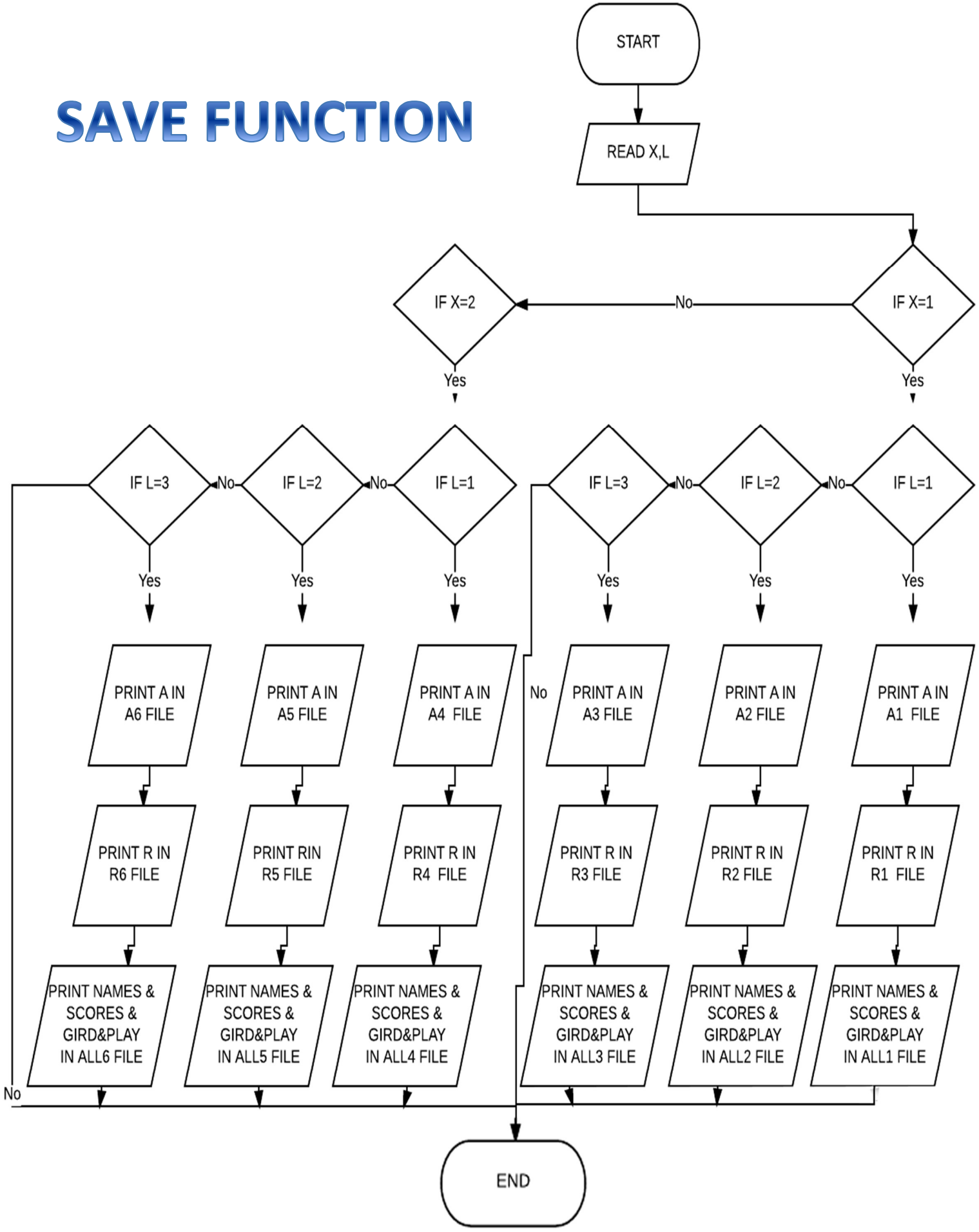
```
((abs(m-
n)==1&&(min!=(grid+1))&&(min!=(grid
+1)*2)&&(min!=(grid+1)*3)&&(min!=(g
rid+1)*4)&&(min!=(grid+1)*5)&&(min!
=(grid+1)*6)&&(min!=(grid+1)*7)&&(m
in!=(grid+1)*8)&&(min!=(grid+1)*9))
||abs(m-n)==(grid+1))&&q==1)
```

Is true , implement following:

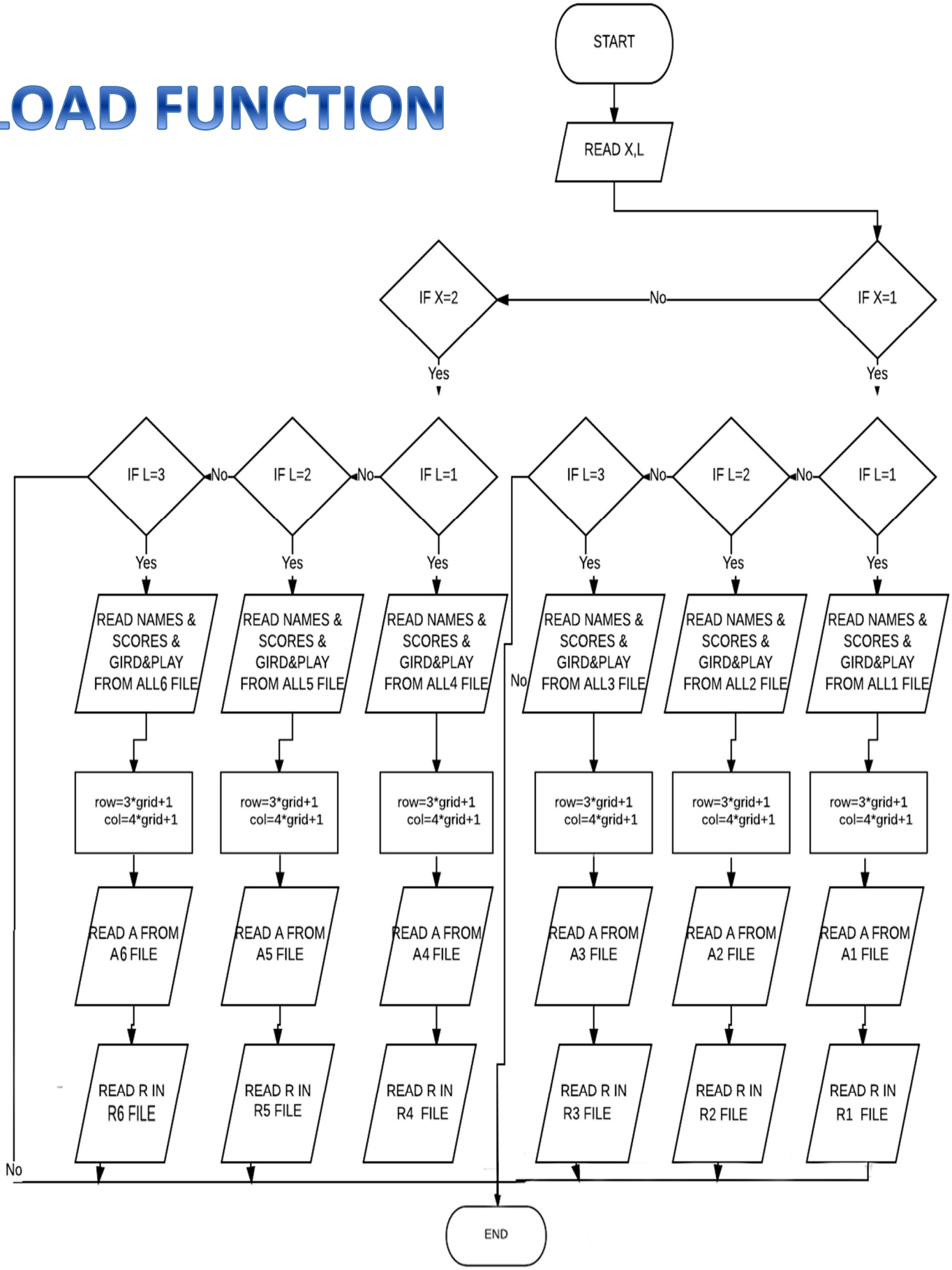
-call boxes function with a , b , n , m , min ,1 as arguments and

register its return value in score variable.
-check if score equal to 1 ,put play equal to 0.
-else if score equal to 2 ,put play equal to 0.
-else if score equal to 0 ,put play equal to 1.
END IF.
-call print_board and digits function.

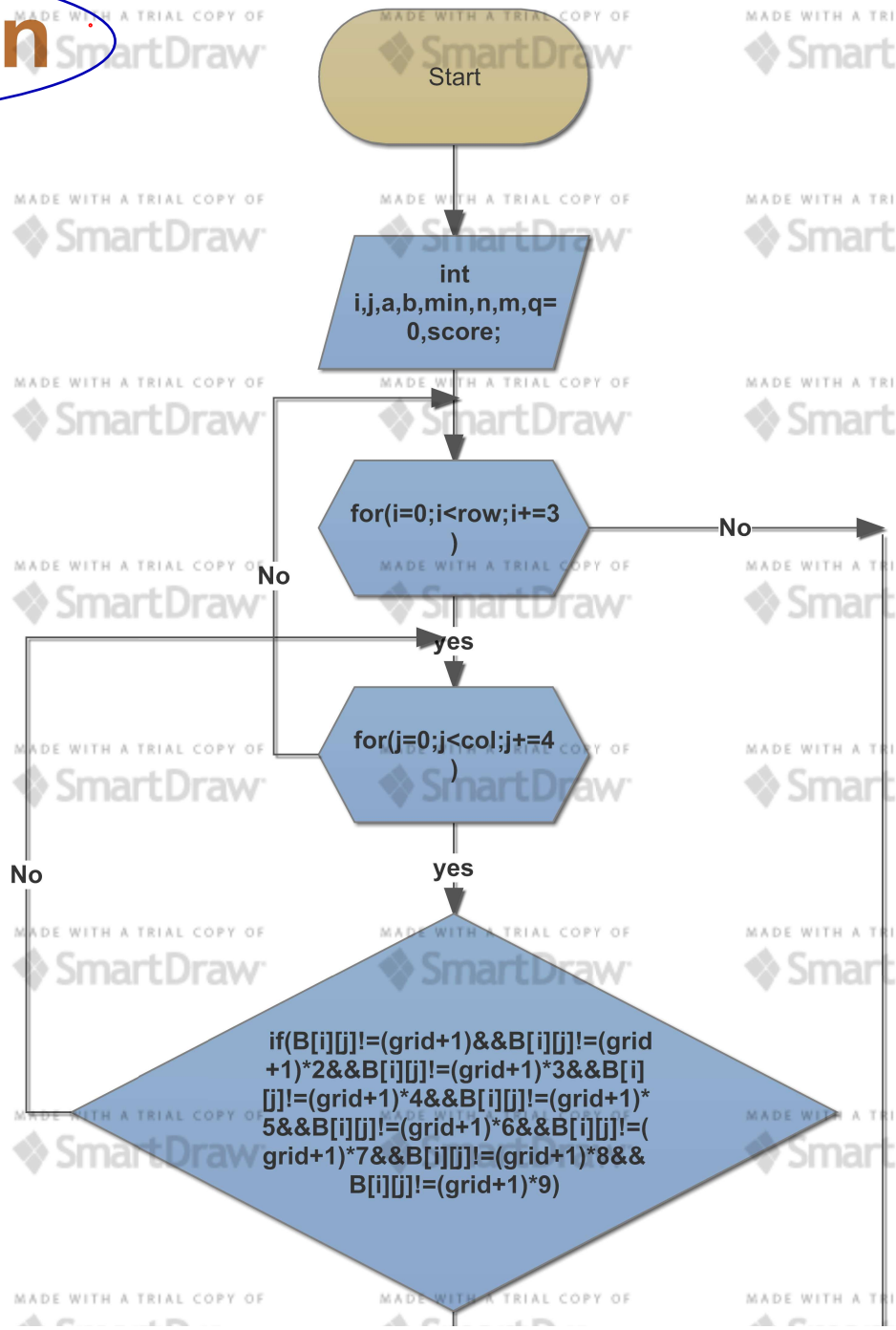
SAVE FUNCTION

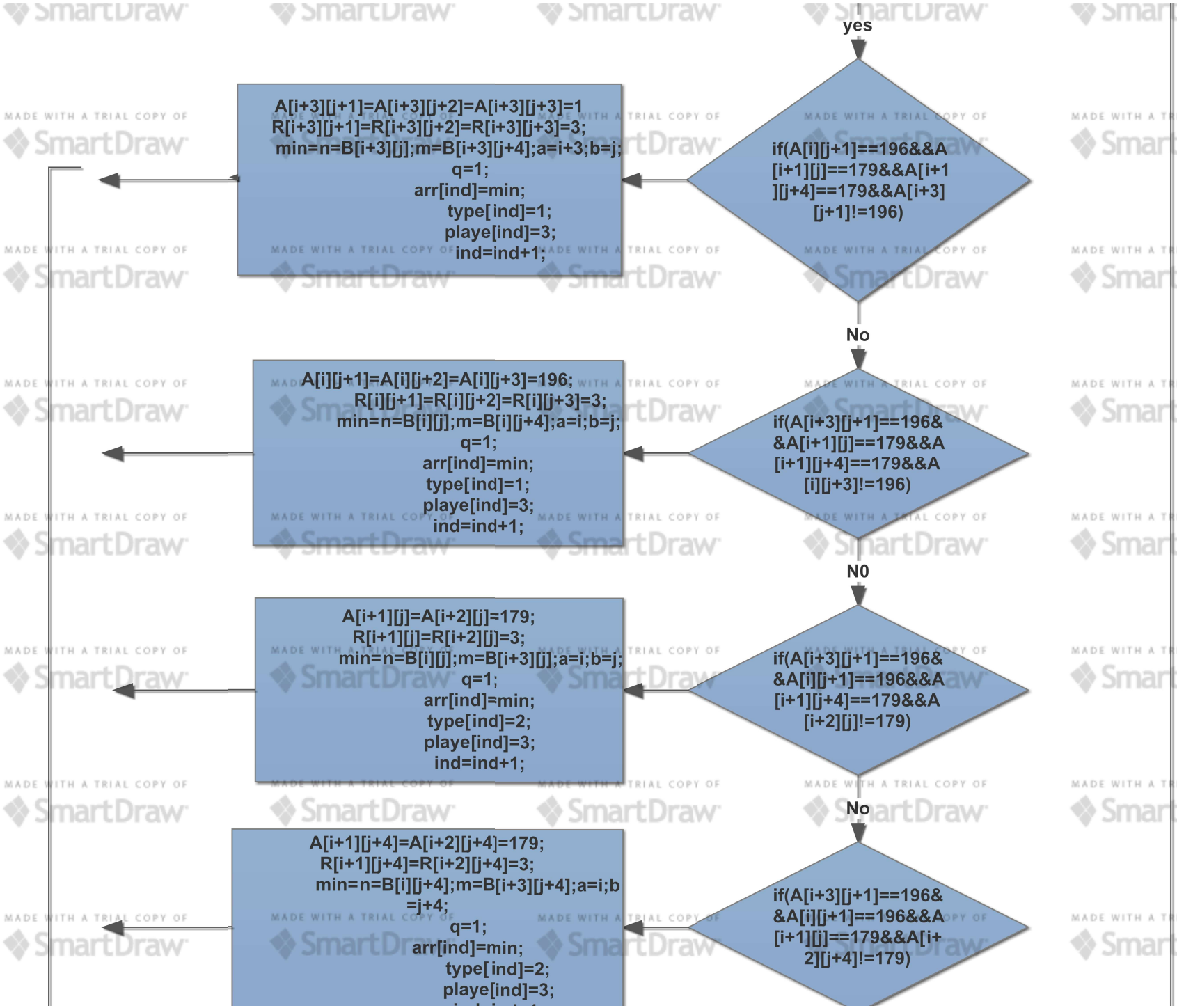


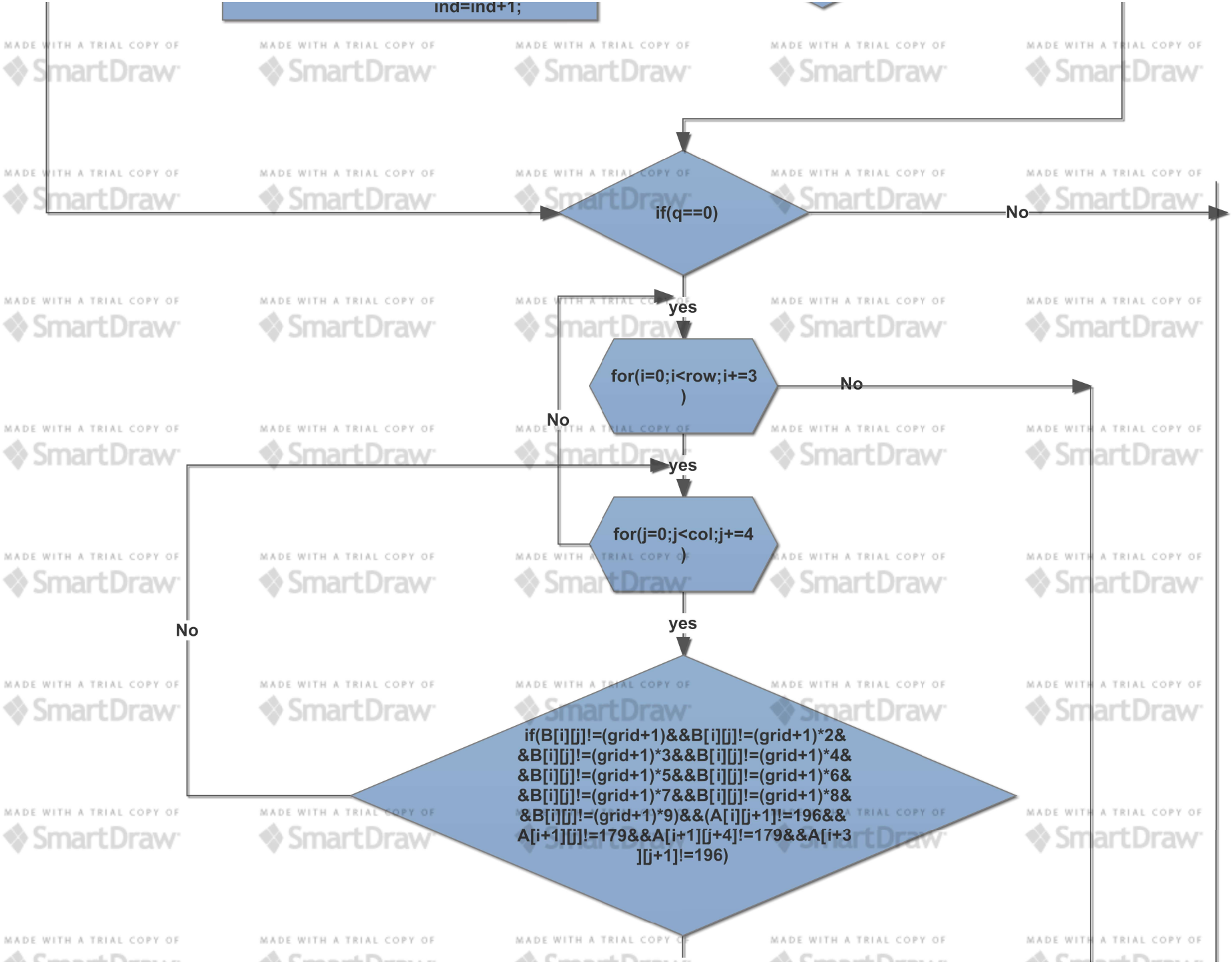
LOAD FUNCTION

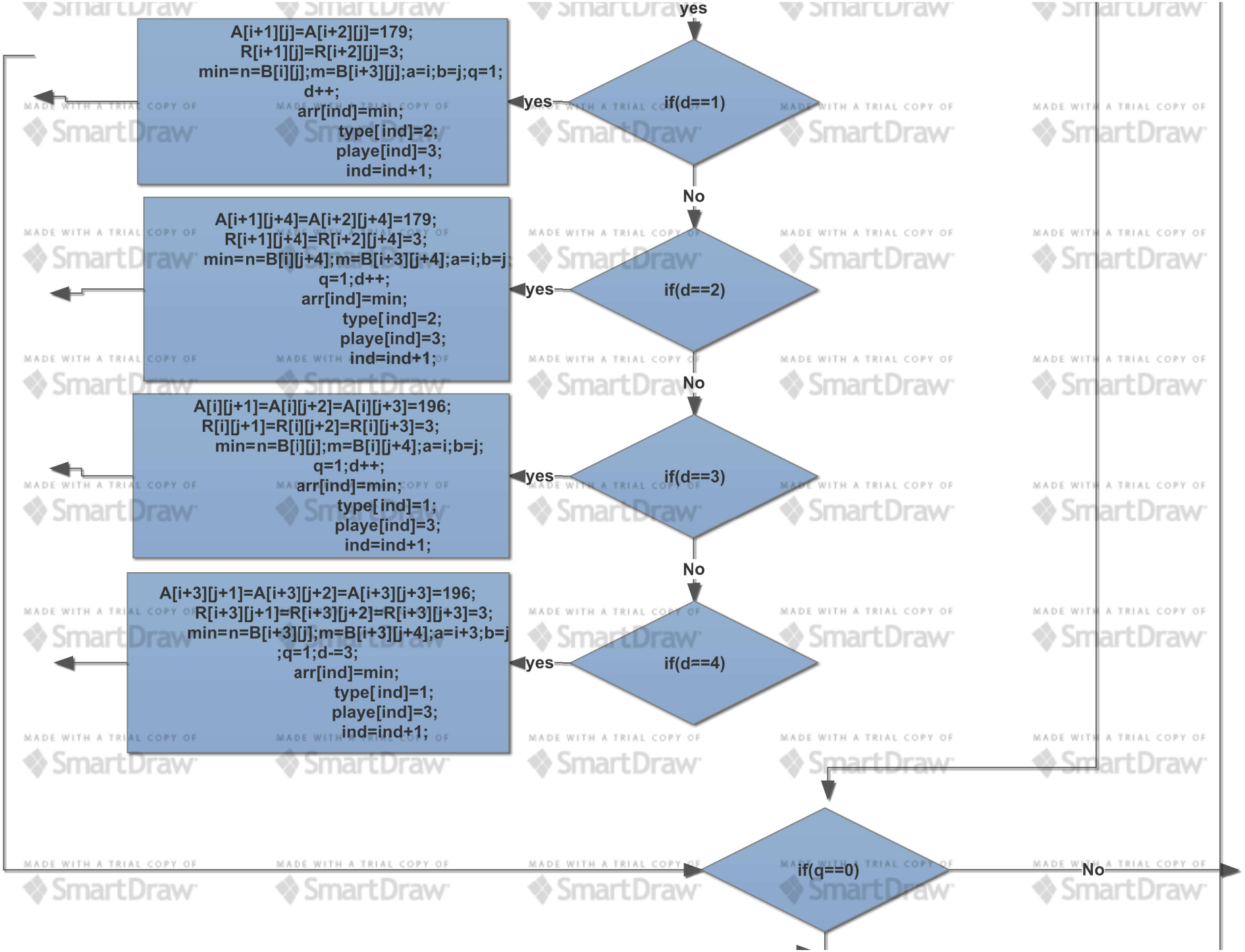


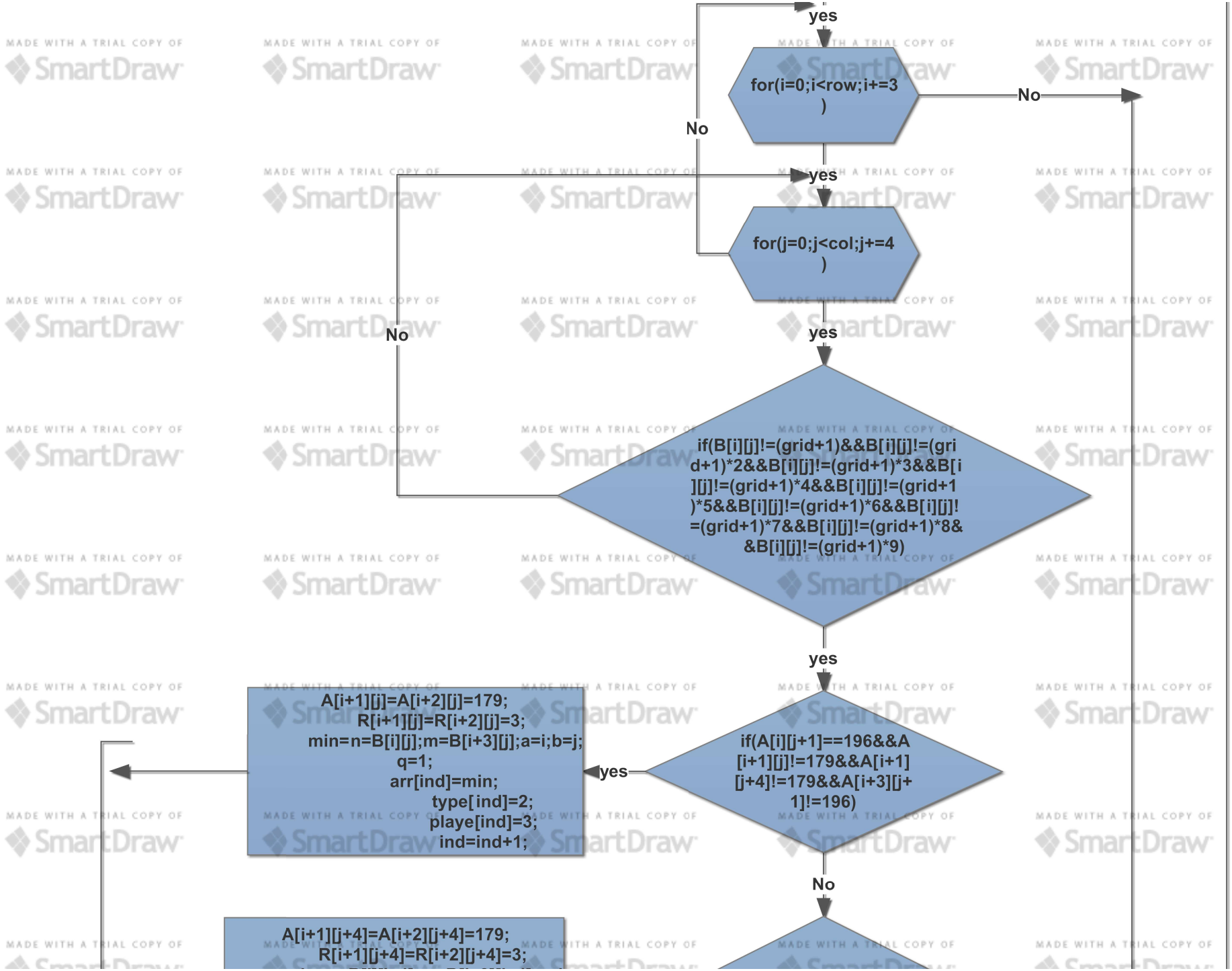
computer function

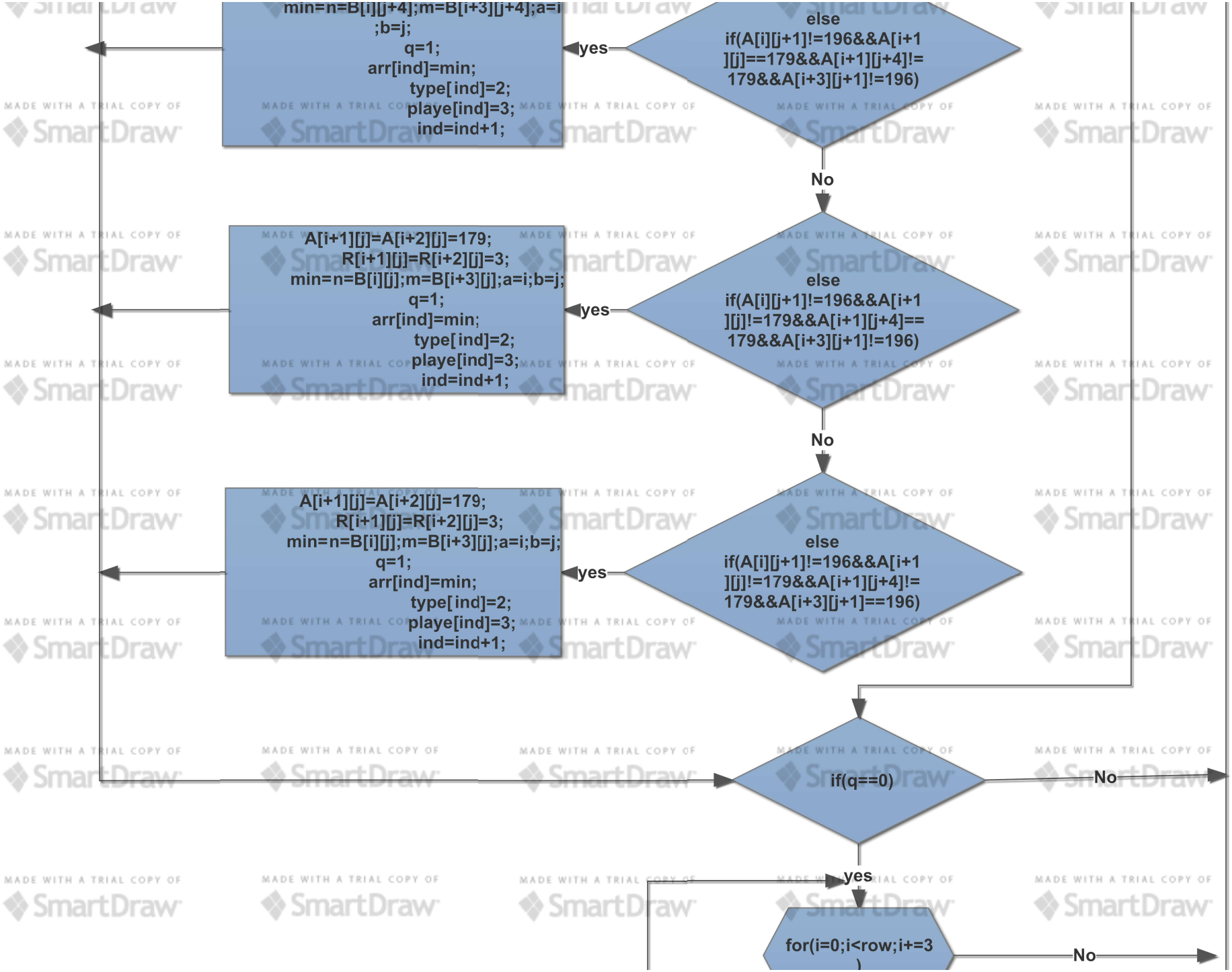


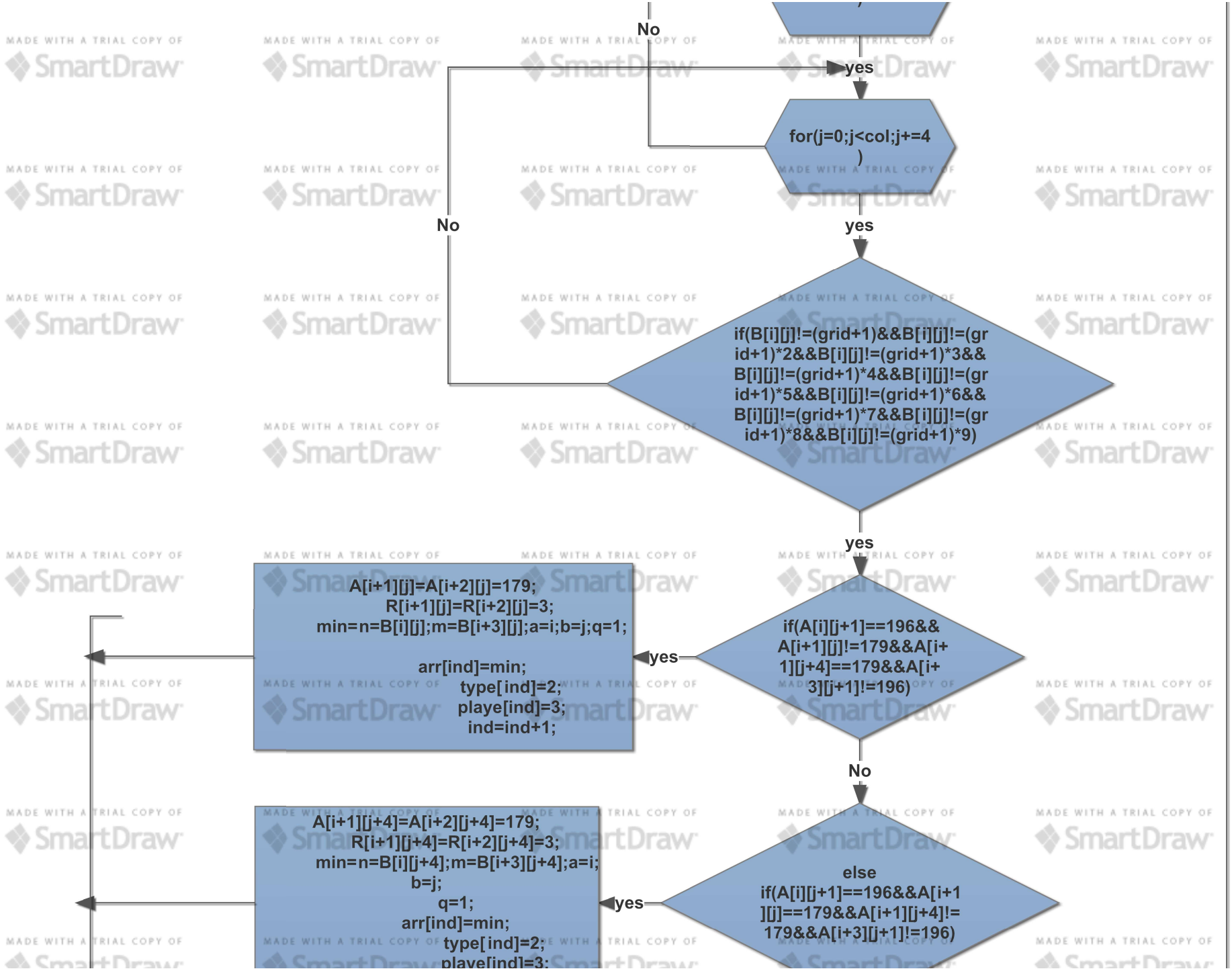

yes
if(A[i][j+1]==196&&A[i+1][j]==179&&A[i+1][j+4]==179&&A[i+3][j+1]!=196)
A[i+3][j+1]=A[i+3][j+2]=A[i+3][j+3]=1
R[i+3][j+1]=R[i+3][j+2]=R[i+3][j+3]=3;
min=n=B[i+3][j];m=B[i+3][j+4];a=i+3;b=j;
q=1;
arr[ind]=min;
type[ind]=1;
playe[ind]=3;
ind=ind+1;
No
if(A[i+3][j+1]==196&&A[i+1][j]==179&&A[i+1][j+4]==179&&A[i][j+3]!=196)
A[i][j+1]=A[i][j+2]=A[i][j+3]=196;
R[i][j+1]=R[i][j+2]=R[i][j+3]=3;
min=n=B[i][j];m=B[i][j+4];a=i;b=j;
q=1;
arr[ind]=min;
type[ind]=1;
playe[ind]=3;
ind=ind+1;
N0
if(A[i+3][j+1]==196&&A[i][j+1]==196&&A[i+1][j+4]==179&&A[i+2][j]!=179)
A[i+1][j]=A[i+2][j]=179;
R[i+1][j]=R[i+2][j]=3;
min=n=B[i][j];m=B[i+3][j];a=i;b=j;
q=1;
arr[ind]=min;
type[ind]=2;
playe[ind]=3;
ind=ind+1;
No
if(A[i+3][j+1]==196&&A[i][j+1]==196&&A[i+1][j]==179&&A[i+2][j+4]!=179)
A[i+1][j+4]=A[i+2][j+4]=179;
R[i+1][j+4]=R[i+2][j+4]=3;
min=n=B[i][j+4];m=B[i+3][j+4];a=i;b=j+4;
q=1;
arr[ind]=min;
type[ind]=2;
playe[ind]=3;

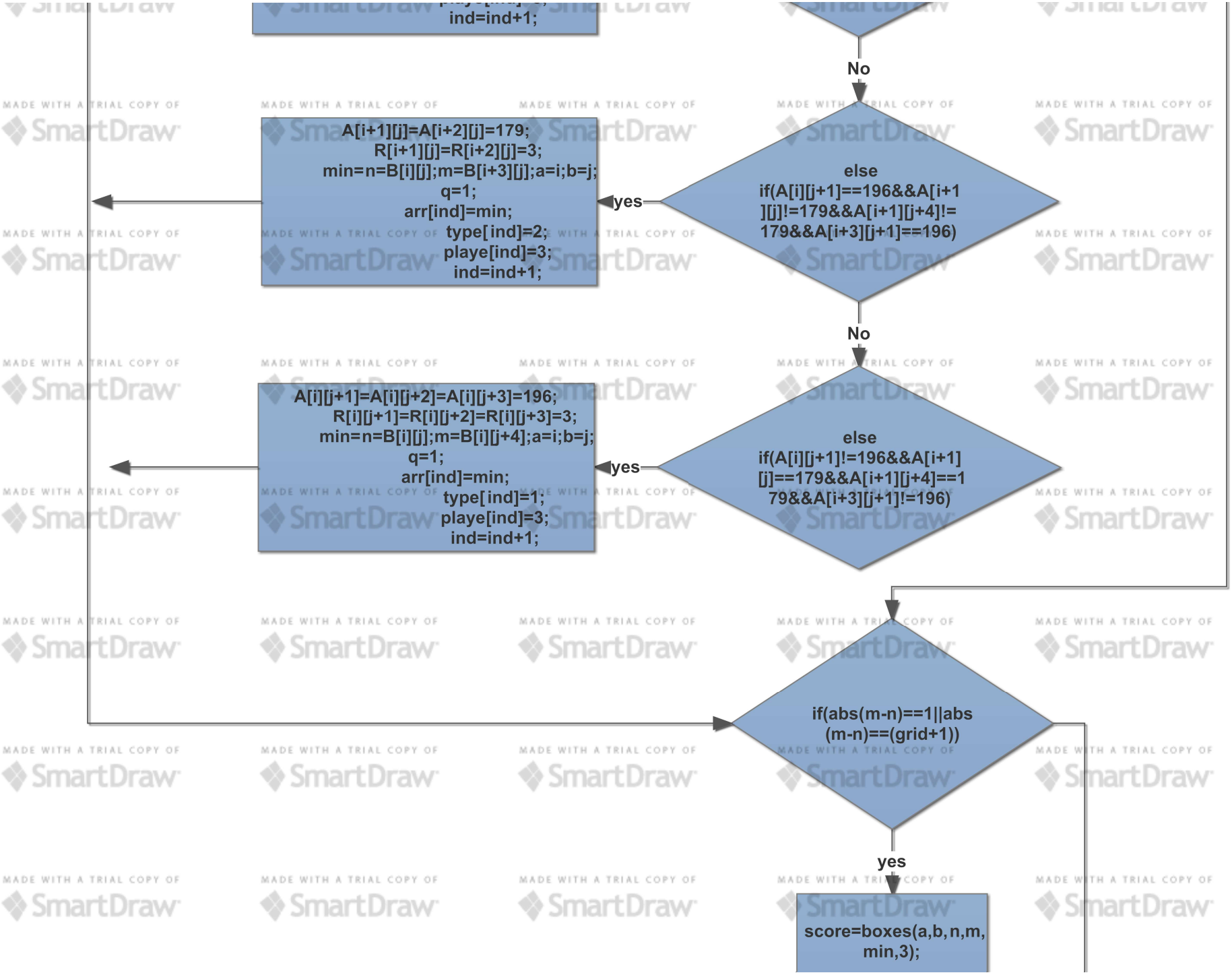

ind=ind+1;
if(q==0)
No
yes
for(i=0;i<row;i+=3
)
No
No
yes
for(j=0;j<col;j+=4
)
yes
No
if(B[i][j]!=(grid+1)&&B[i][j]!=(grid+1)*2&
&B[i][j]!=(grid+1)*3&&B[i][j]!=(grid+1)*4&
&B[i][j]!=(grid+1)*5&&B[i][j]!=(grid+1)*6&
&B[i][j]!=(grid+1)*7&&B[i][j]!=(grid+1)*8&
&B[i][j]!=(grid+1)*9)&&(A[i][j+1]!=196&&
A[i+1][j]!=179&&A[i+1][j+4]!=179&&A[i+3
][j+1]!=196)

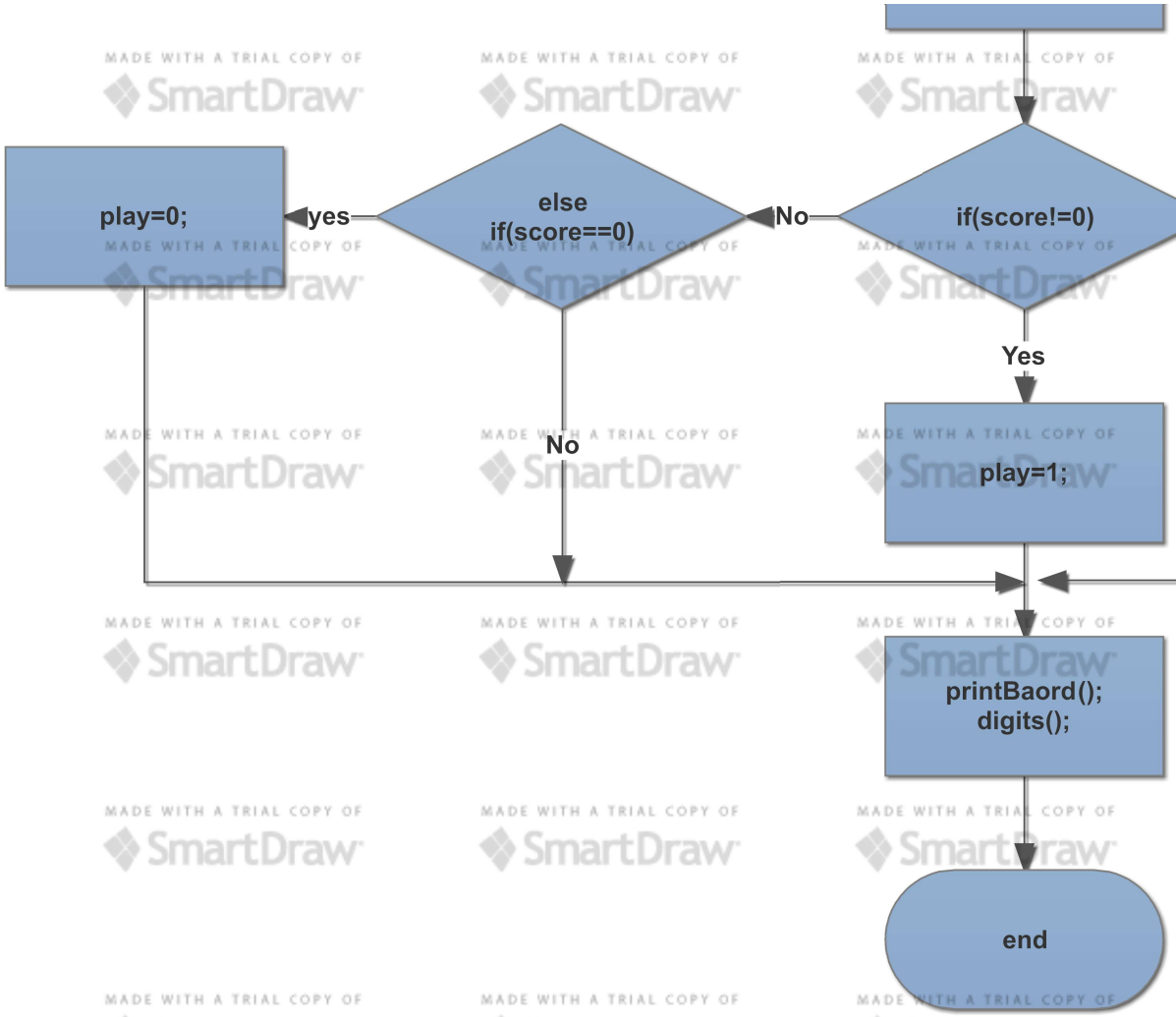

yes
if(d==1)
yes
A[i+1][j]=A[i+2][j]=179;
R[i+1][j]=R[i+2][j]=3;
min=n=B[i][j];m=B[i+3][j];a=i;b=j;q=1;
d++;
arr[ind]=min;
type[ind]=2;
playe[ind]=3;
ind=ind+1;
No
if(d==2)
yes
A[i+1][j+4]=A[i+2][j+4]=179;
R[i+1][j+4]=R[i+2][j+4]=3;
min=n=B[i][j+4];m=B[i+3][j+4];a=i;b=j;
q=1;d++;
arr[ind]=min;
type[ind]=2;
playe[ind]=3;
ind=ind+1;
No
if(d==3)
yes
A[i][j+1]=A[i][j+2]=A[i][j+3]=196;
R[i][j+1]=R[i][j+2]=R[i][j+3]=3;
min=n=B[i][j];m=B[i][j+4];a=i;b=j;
q=1;d++;
arr[ind]=min;
type[ind]=1;
playe[ind]=3;
ind=ind+1;
No
if(d==4)
yes
A[i+3][j+1]=A[i+3][j+2]=A[i+3][j+3]=196;
R[i+3][j+1]=R[i+3][j+2]=R[i+3][j+3]=3;
min=n=B[i+3][j];m=B[i+3][j+4];a=i+3;b=j
;q=1;d-=3;
arr[ind]=min;
type[ind]=1;
playe[ind]=3;
ind=ind+1;
if(q==0)
No


yes
for(i=0;i<row;i+=3
)
No
No
yes
for(j=0;j<col;j+=4
)
No
yes
if(B[i][j]!=(grid+1)&&B[i][j]!=(gri
d+1)*2&&B[i][j]!=(grid+1)*3&&B[i
][j]!=(grid+1)*4&&B[i][j]!=(grid+1
)*5&&B[i][j]!=(grid+1)*6&&B[i][j]!
=(grid+1)*7&&B[i][j]!=(grid+1)*8&
&B[i][j]!=(grid+1)*9)
yes
A[i+1][j]=A[i+2][j]=179;
R[i+1][j]=R[i+2][j]=3;
min=n=B[i][j];m=B[i+3][j];a=i;b=j;
q=1;
arr[ind]=min;
type[ind]=2;
playe[ind]=3;
ind=ind+1;
yes
if(A[i][j+1]==196&&A
[i+1][j]!=179&&A[i+1]
[j+4]!=179&&A[i+3][j+
1]!=196)
No
A[i+1][j+4]=A[i+2][j+4]=179;
R[i+1][j+4]=R[i+2][j+4]=3;


min=n=B[i][j+4];m=B[i+3][j+4];a=i
;b=j;
q=1;
arr[ind]=min;
type[ind]=2;
playe[ind]=3;
ind=ind+1;
yes
else
if(A[i][j+1]!=196&&A[i+1
][j]==179&&A[i+1][j+4]!=
179&&A[i+3][j+1]!=196)
No
A[i+1][j]=A[i+2][j]=179;
R[i+1][j]=R[i+2][j]=3;
min=n=B[i][j];m=B[i+3][j];a=i;b=j;
q=1;
arr[ind]=min;
type[ind]=2;
playe[ind]=3;
ind=ind+1;
yes
else
if(A[i][j+1]!=196&&A[i+1
][j]!=179&&A[i+1][j+4]==
179&&A[i+3][j+1]!=196)
No
A[i+1][j]=A[i+2][j]=179;
R[i+1][j]=R[i+2][j]=3;
min=n=B[i][j];m=B[i+3][j];a=i;b=j;
q=1;
arr[ind]=min;
type[ind]=2;
playe[ind]=3;
ind=ind+1;
yes
else
if(A[i][j+1]!=196&&A[i+1
][j]!=179&&A[i+1][j+4]!=
179&&A[i+3][j+1]==196)
if(q==0)
No
yes
for(i=0;i<row;i+=3
)
No


No
yes
for(j=0;j<col;j+=4
)
No
yes
if(B[i][j]!=(grid+1)&&B[i][j]!=(gr
id+1)*2&&B[i][j]!=(grid+1)*3&&
B[i][j]!=(grid+1)*4&&B[i][j]!=(gr
id+1)*5&&B[i][j]!=(grid+1)*6&&
B[i][j]!=(grid+1)*7&&B[i][j]!=(gr
id+1)*8&&B[i][j]!=(grid+1)*9)
yes
A[i+1][j]=A[i+2][j]=179;
R[i+1][j]=R[i+2][j]=3;
min=n=B[i][j];m=B[i+3][j];a=i;b=j;q=1;
arr[ind]=min;
type[ind]=2;
playe[ind]=3;
ind=ind+1;
yes
if(A[i][j+1]==196&&
A[i+1][j]!=179&&A[i+
1][j+4]==179&&A[i+
3][j+1]!=196)
No
A[i+1][j+4]=A[i+2][j+4]=179;
R[i+1][j+4]=R[i+2][j+4]=3;
min=n=B[i][j+4];m=B[i+3][j+4];a=i;
b=j;
q=1;
arr[ind]=min;
type[ind]=2;
playe[ind]=3;
yes
else
if(A[i][j+1]==196&&A[i+1
][j]==179&&A[i+1][j+4]!=
179&&A[i+3][j+1]!=196)


ind=ind+1;
No
A[i+1][j]=A[i+2][j]=179;
R[i+1][j]=R[i+2][j]=3;
min=n=B[i][j];m=B[i+3][j];a=i;b=j;
q=1;
arr[ind]=min;
type[ind]=2;
playe[ind]=3;
ind=ind+1;
yes
else
if(A[i][j+1]==196&&A[i+1
][j]!=179&&A[i+1][j+4]!=
179&&A[i+3][j+1]==196)
No
A[i][j+1]=A[i][j+2]=A[i][j+3]=196;
R[i][j+1]=R[i][j+2]=R[i][j+3]=3;
min=n=B[i][j];m=B[i][j+4];a=i;b=j;
q=1;
arr[ind]=min;
type[ind]=1;
playe[ind]=3;
ind=ind+1;
yes
else
if(A[i][j+1]!=196&&A[i+1]
[j]==179&&A[i+1][j+4]==1
79&&A[i+3][j+1]!=196)
if(abs(m-n)==1||abs
(m-n)==(grid+1))
yes
score=boxes(a,b,n,m,
min,3);



Boxes function