**Important Parameters**

* **Turn parameter :**

to reverse the turn when we need to check on the pieces of the other player

* **r1 , c1 ,r2 ,c2 parameters :**

the place at which the piece exist and the other place we want to move to

* **rk ,ck ,Rk,cK (global):**

holds the place of the two kings

* **moves [1000][7] (global):**

saves any move I make

* **place[8][8] (global):**

holds the place of each piece .

* **place colors (global) :**

to hold if these place has \_ or .

* **white [8] , black [8] (global):**

**holds white and black pieces .**

* **rowlast , columnlast(global) :**

the place of the piece threating thee king

**Important Functions**

* **SetColor(Taken from The Internet) :**

Changes the color of the text in CMD

* **Printc :**

Outputs the pored and its pieces on the screen

* **Clearc :**

Removes the extra entries after space so the game don’t skip a turn

* **placef :**

**moves a piece from r1,c1 to r2,c2 .**

* **Checkf :**

Checks if the desired piece belongs to the color of the player which has the turn to play or not .

* **invalid :**

if the move was invalid it sacns for new move

* **promotion :**

­ **see if theres a fifth charcter entered and promotes the pawn to it**

* **undo :**

undone a move .

* **redo :**

redone a move .

* **checkmatef :**

**checks if someone is threaating the king**

**algorithm :**

**start**

**initialize I , j ;**

**for (i=0 ; i<=7 ; i++){**

**for (j=0 ; j<=7 ; j++){**

**if (turn is white){**

**switch(the piece exists at place[i][j]) {**

**see if this piece is threating the white king and**

**and returns 1 if there was**

**}else {**

**See if this piece is threating the black king and returns 1 if there was**

**}}**

**}**

**If (any piece can move to king position ){**

**Print “check”}else return 0**

End

* **last check :**

if checkmatef returned 1 , checks if any piece from the opponent can kill the threating piece or blocks its path .

algorithm :

start

initialize I , j ,k, rk , ck , rK, cK

calculate difference = distance between the piece threating my king and king place

for(k=0;k<difference;k++){

**for (i=0 ; i<=7 ; i++){**

**for (j=0 ; j<=7 ; j++){**

if( my piece can move to place[rowlast][columnlast]){

return 1}

}}

Add or subtract one (rowlast , columnlast)

}

Return 0 ; // if no one can move

End

* **winnerf :**

checks if the king can move in any place around him without being threatened , if he can’t thin the opponent wins .

* **stalecheck :**

checks if there is a checkmate on the king after removing one of my pieces

* **Stalemate :**

Checks if my king can move to any place around him without being checked if not print stalemate and exit the game

* **Possiple :**

**Checks if any of my pieces can make any move without making a checkmate on the king**

**Algorithm :**

**Start**

**Initialize I , j , m , n ,value , checkmate**

**for (i=0 ; i<=7 ; i++){**

**for (j=0 ; j<=7 ; j++){**

**see which piece exist at place[i][j]**

**for(m ; i+m <8 && m <2 ; m++){**

**for (n ; j+n < 8 && n < 2 ;n++){**

**value = move piece at place[i][j] to place[m][n]**

if(value =1 ) {

checkmate = checkmatef()

if (checkmate = 0){

return 0 }

}}}}}

Return 1

End

**Main**

**Algorithm :**

Start

Initialize fail =1

While (True){

Here : If ( fail =0 ){

Invalid();}

Else {

system("cls");

checkmatef(); // to print check if there was

Read in move;

If (move = ‘S’)

Save (place , graveyard , moves , count)

Else if( move = ‘L’)

Load (place , graveyard , moves , count)

Else if (move = ‘U’)

Undo();

Else if (move = ‘R’)

Redo();

Else {

check which piece at place ;

}

If (fail =1 ){

**Checkmate=Checkmatef() ;**

**// checks my move had caused any threat to my king or not**

**If(checkmate = 1){**

**Undo() ;**

**Goto here ;}}**

**If (fail =1 ){**

**Checkmate = checkmatef();**

**// check if I caused threat on opponent king**

**Lastcheck = Lastcheckf();**

**Poss = possible();**

**If ( poss =1 and checkmate = 0 ){**

**Stalemate();}**

**If(last check = 0and checkmate =1){**

**Winner() ;**

**}**

**Save Move ;**

**}**

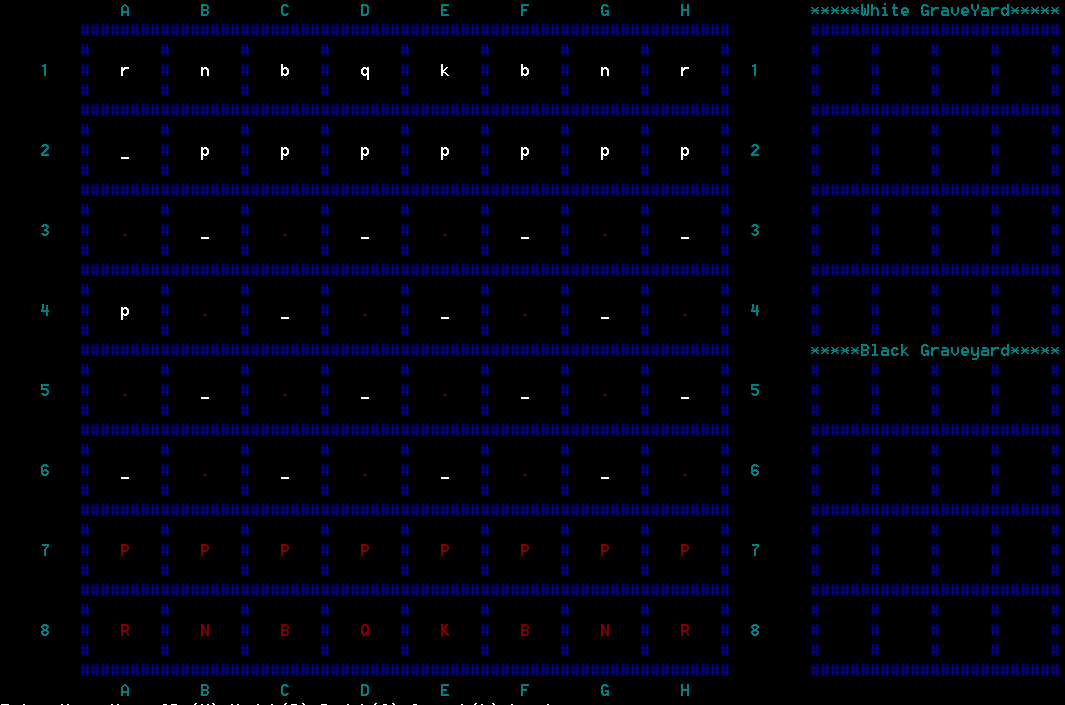
**Return 0 ;**

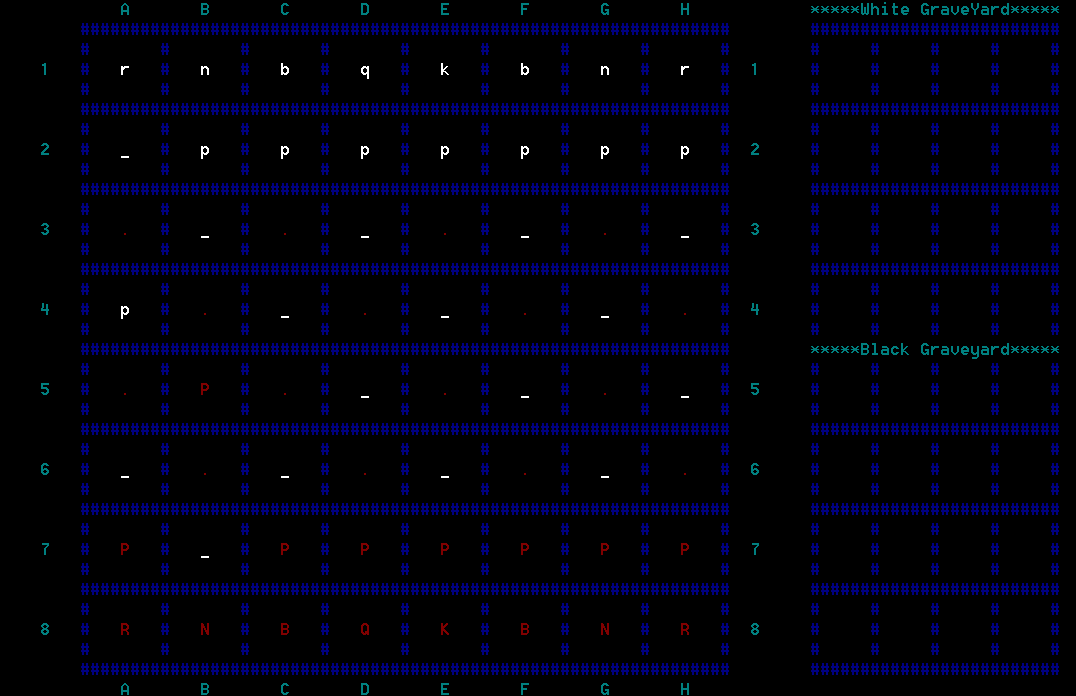
**}**

**end**

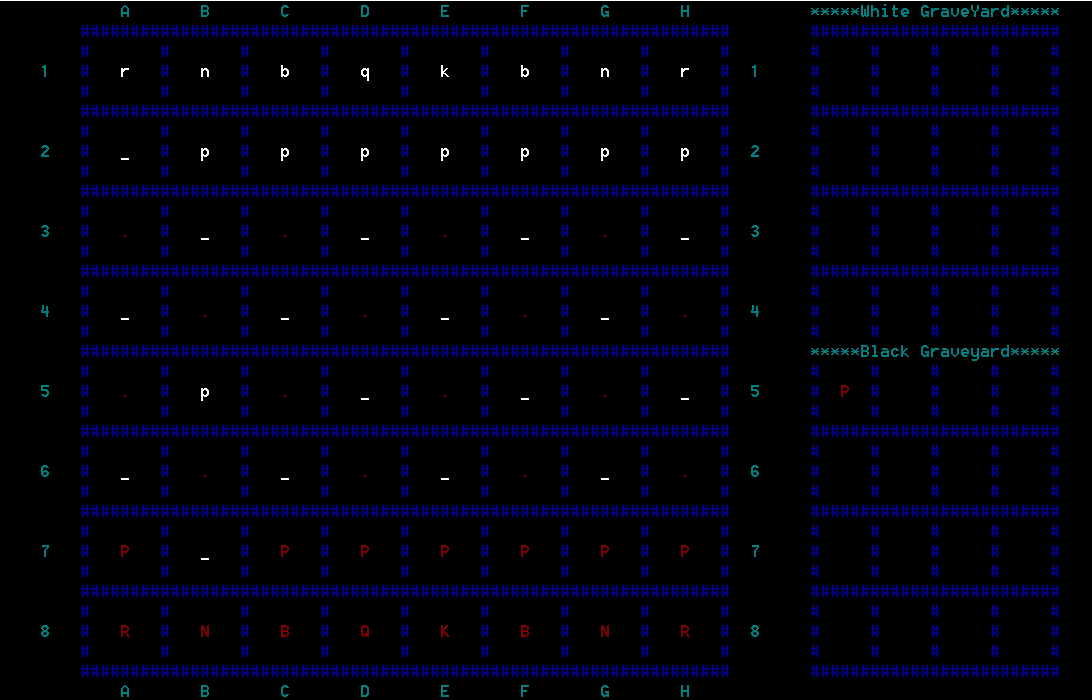
**Simple Moves**

**Normal move :**

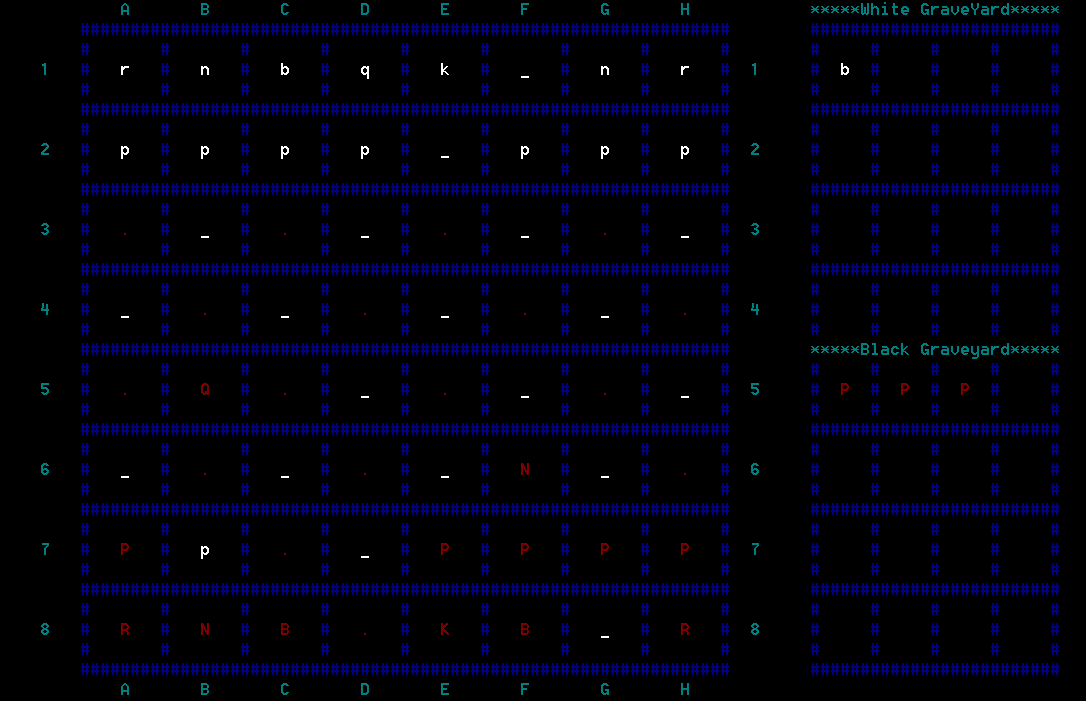
1. **A2A4**

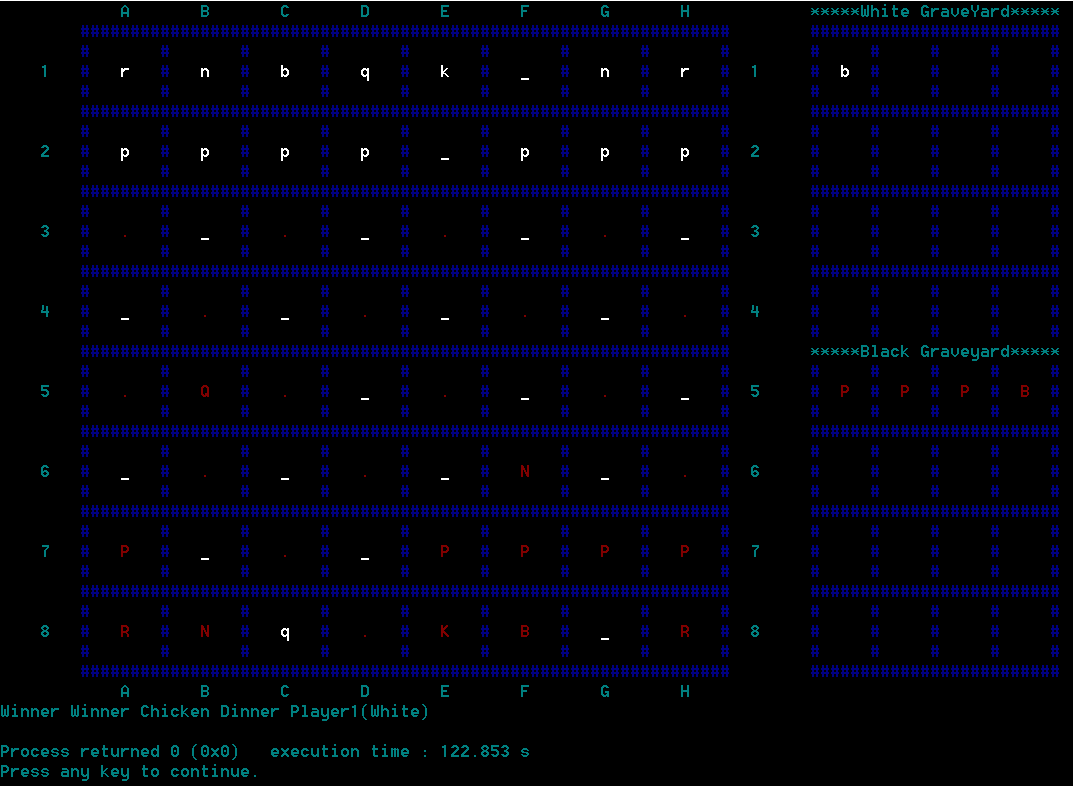
** 2)B7B5**

**Eating :**

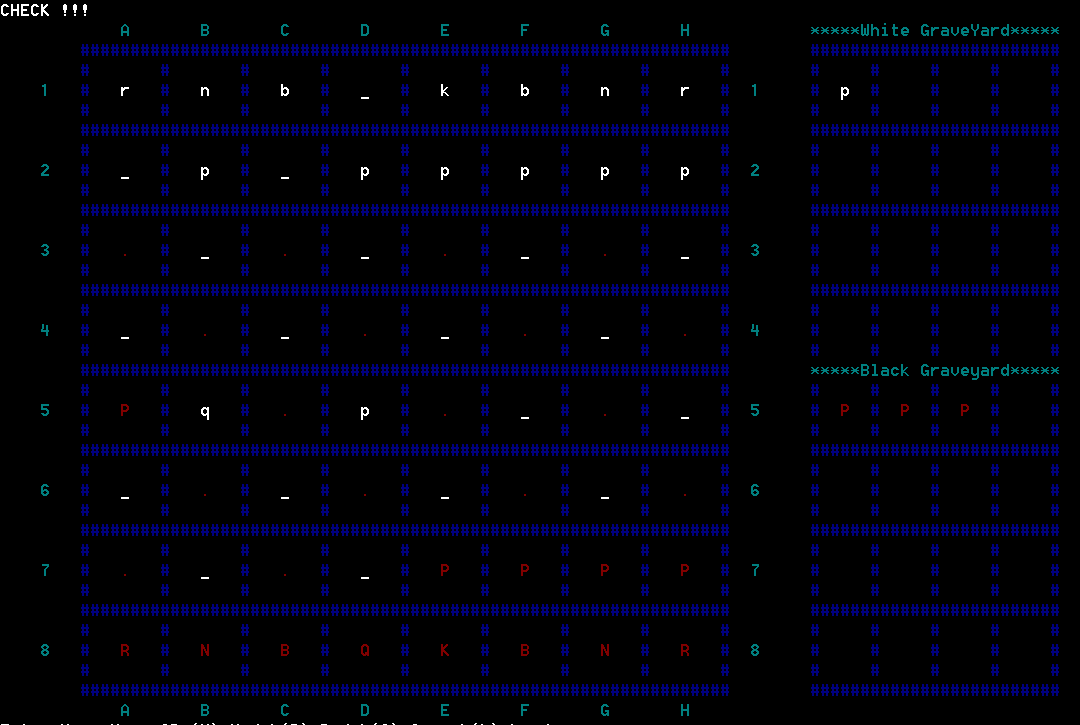
**A4B5**

**Promotion :**

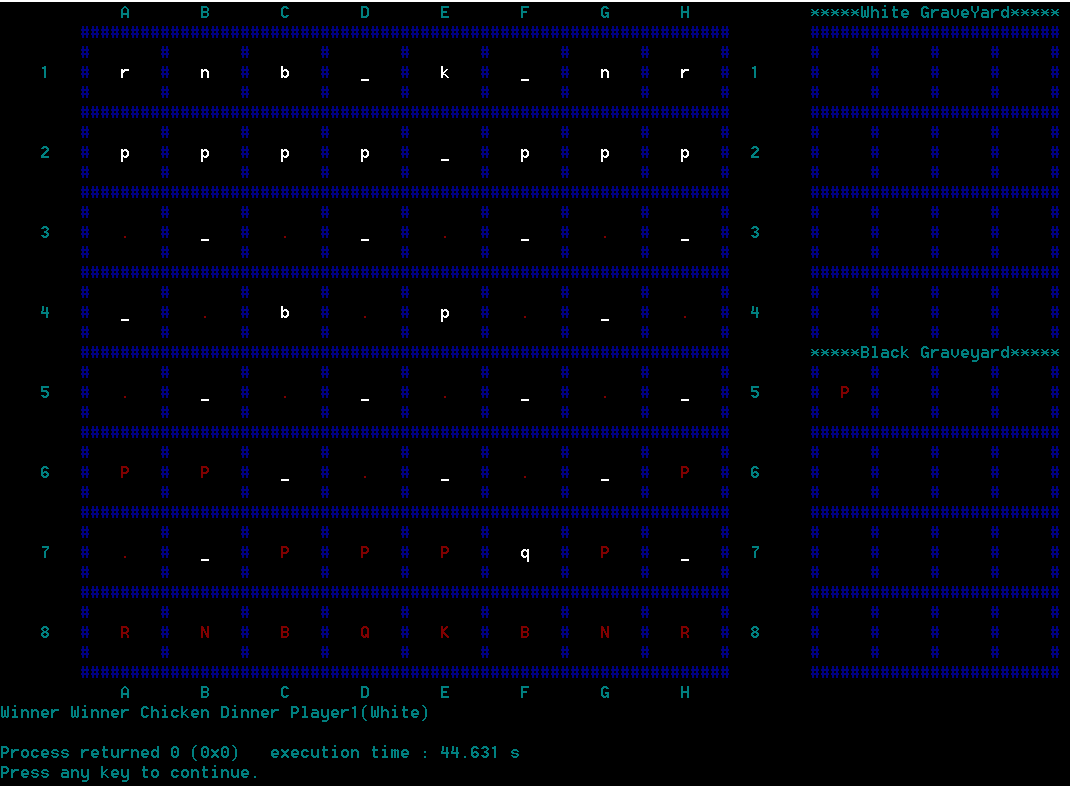
****

 B7C8Q(Eating bishop and promotion to queen and checkmate)

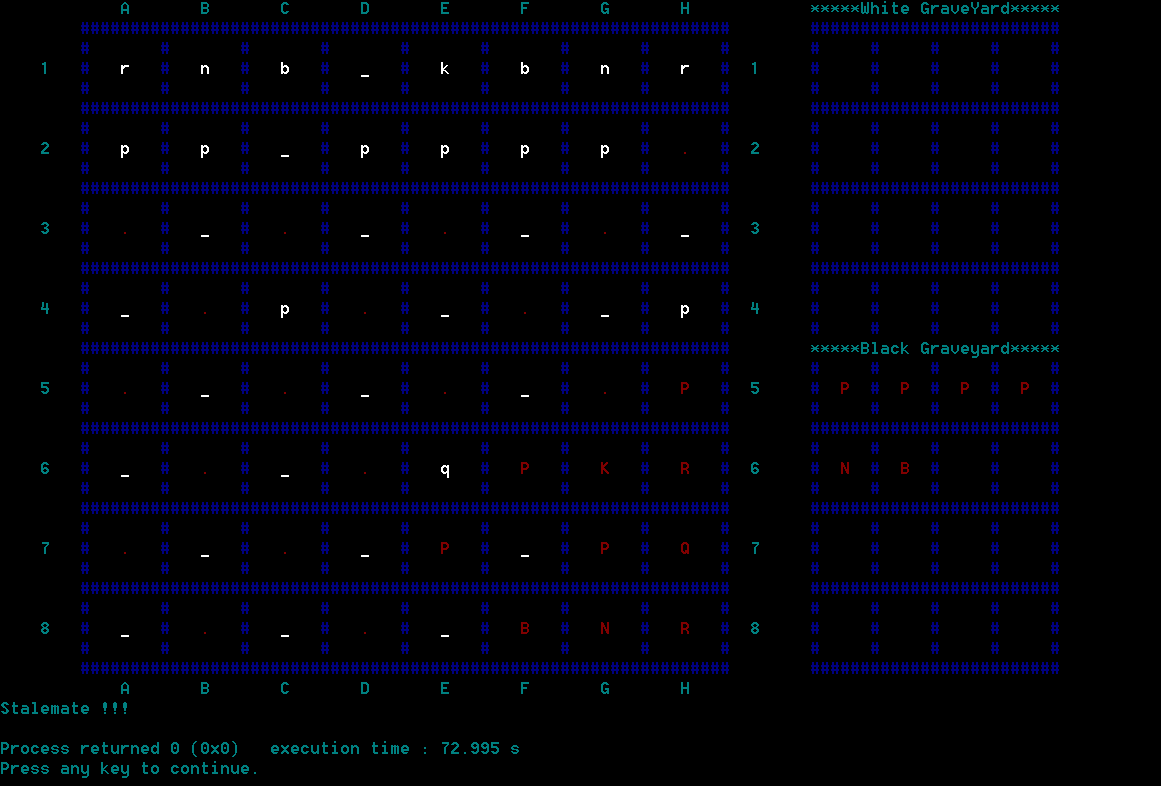
**Check :**

** A4B5(QUEEN)**

**Checkmate :**

** H5F7**

**Stalemate :**

**C8E8**