```
#include <stdio.h>
#include <stdbool.h>
#include <curses.h>
#include <unistd.h>
#define ASCII_ESC 27


struct bdt {
int b[2][7];
int w[2];
};

struct mvt {
struct bdt board;
bool e;
};
struct evt {
int val;
int ii;
};

struct lct {
int c;
int l;
};

struct bdt board;
int val,mv,c,ii,i,j,n,basec,basel,incc,incl,dummy,go;
bool pl,e,reach,validmv;
//long int noise;
char ch;
int run;
int x,y,l;
//char *string;

void CLRSCRN (void);
void PBDSKEL (void);
void PBDVAL (void);

void PLAY (int ci,int s);

struct evt EVAL (int c,int n,struct bdt aboard);

void CUP (int ll,int cc);

int main ()

{
/*start of main program*/
        //TextBackground (black);
```

```
        //ClrScr;
        //TextColor(white);
        struct mvt breturn;
        struct evt evalret;

        CLRSCRN();
        printf("                          An African Game -- OWARE
v1.0.rev2002\n");
        printf("Would like to play a game? [No==0, Yes==1]");
        scanf ("%1d",&dummy);

        while (dummy == 1)  {
                for (i=1; i<= 6;i++) {
                        board.b[0][i]=4;
                        board.b[1][i]=4;
                        }
                board.w[0]=0;
                board.w[1]=0;
                printf("Do you like to read instructions? [No=0,
Yes=1]");
        scanf("%1d",&mv);
        if (mv==1) {
                printf("\n");
                printf("The game is played in Africa by digging 12
holes etc.......\n");
                printf("\n");
                }
        printf("who goes first ? [ computer==0, player==1] ");
        scanf("%1d",&c);

        //printf ("c=%d",c);

        basec=5;
        basel=10;
        incc=10;
        incl=6;     /*  printf("Input Level ?");*/
        //read (n);
//        }
        n=3;//look 3 moves ahead to chose min/max best move
        e=false;
        //pl=false;
        CLRSCRN();

        PBDSKEL();

        PBDVAL();
        while (! e ) {
                if (c==1)  {//play user as computer in duel

                        CUP(3,40);//TextColor(magenta);
```

```
                        printf("user playing");

                        evalret=EVAL (c,n,board);//plus call
                        val=evalret.val;
                        ii=evalret.ii;

                        //pl=true;
                        CUP(3,1);
                        printf("user moved:");
                        //Str(7-ii:4,s);
                        printf("%4d", ii);
                        CUP(21,1);
                        printf("type 1<cr> to play the user move: ");

                        do scanf ("%d",&go); while (go != 1);

                        //run=1; /*actual play move*/
                        CUP(3,1);
                        printf("                    ");

                        PLAY(c,ii);//sets a global e to end

                        CUP(3,40);
                        printf("
");
                        CUP(21,1);
                        printf("
");


                        }
                else
                        {
                        CUP(3,40);//TextColor(magenta);
                        printf("computer playing");

                        evalret=EVAL (c,n,board);//plus call
                        val=evalret.val;
                        ii=evalret.ii;

                        //pl=true;
                        CUP(3,1);
                        printf("computer moved:");
                        //Str(7-ii:4,s);
                        printf("%4d", 7-ii);
                        CUP(21,1);
                        printf("type 1<cr> to play the computer move:
");

                        do scanf ("%d",&go); while (go != 1);
```

```
                        //run=1; /*actual play move*/
                        CUP(3,1);
                        printf("                    ");

                        PLAY(c,ii);//sets a global e to end

                        CUP(3,40);
                        printf("
");
                        CUP(21,1);
                        printf("
");
                        }
                //PBDVAL(board);
                //pl=false;
                //c=1-c;

                if (c==0) {c=1;} else {c=0;}

                }//end wjile !e

        CUP(22,1);//TextColor(white);ClrEol;
        printf("would you like to play another game? [No==0,
Yes==1]");
        scanf ("%1d",&dummy);
        }

}

void CUU (int n)
        {
// ^[[<n>A
        printf( "%c[%dA", ASCII_ESC, n );
       }
void CUB (int n)
        {
// ^[[<n>D
        printf( "%c[%dD", ASCII_ESC, n );
        }
void CUD( int n)
        {
// ^[[<n>B
        printf( "%c[%dB", ASCII_ESC, n );
        }
void CUP (int ll,int cc)
        {
// ^[[<v>;<h>H
        printf( "%c[%d;%dH", ASCII_ESC,ll,cc );
        }
void CLRSCRN (void)
```

```
        {
        printf ("\033[2J");
        }

struct lct GETLC(int r, int c0, int l0)
        {
        int u,t,v1,v2,j,k,c,l;
        struct lct getlcret;

        u=r / 16;
        t=r % 16;
        v1=t / 8;
        t=t % 8;
        v2=t / 4;
        t=t % 4;
        j=t / 2;
        k=t % 2;
        c=c0+4*u+2*v2+j;
        l=l0-2*v1-k;

        getlcret.c=c;
        getlcret.l=l;

        return getlcret;
        }

void PHOLE(int pi,int pj,int m)
        {
        int c0,l0,l,c,r,u,t,v1,v2,j,k;
        struct lct getlcret;

        c0=basec+(pj-1)*incc;
        l0=basel+pi*incl;

        for (r=1;r<= 32;r++) {
                getlcret=GETLC(r,c0,l0);
                l=getlcret.l;
                c=getlcret.c;

                CUP(l,c);
                printf(" ");
                //usleep(5000);
                }

        //usleep(1000000);

        if (m>0) {
        for (r=1;r<= m;r++) {
                getlcret=GETLC(r,c0,l0);
                l=getlcret.l;
```

```
                c=getlcret.c;


                CUP(l,c);
                //printf("%s",pch);
                printf("%s","*");
                //usleep(5000);
                //sleep(10);
                }

             //sleep(1000);
            }

        //usleep(500000);
        //sleep(1);
        }

/*void PVAL(int i, int j, int m)
        {
//      const delayval = 400;
        int x,y,oldtm,newtm;
//        TextColor(green);
//      m=board.b[i][j];
//      if (m!=0) {PHOLE(i,j,m,"*");}
        PHOLE(i,j,m);
//        Delay(delayval);
        }
*/

void PWVAL (int w0, int w1)
        {
        CUP(1,1);
        //TextColor(red);
        printf("computer has won ");
        //Str(board.w[0]:5,s);
        printf("%d",w0);
//        printf("%d",board.w[0]);
        printf("  pebbles\n");
        printf("user has won ");
        //Str(board.w[1]:5,s);
        printf("%d",w1);
        printf(" pebbles\n");
        }

void PBDVAL (void)
        {
        int i,j,w0,w1,m;
        for (i=0;i<=1;i++) {
                for (j=1;j<=6;j++) {
                        m=board.b[i][j];
```

```
                        PHOLE(i,j,m);
                        //if (m!=0) {PHOLE(i,j,m,"*");}

                        }
                }
        w0=board.w[0];
        w1=board.w[1];
        PWVAL(w0,w1);
        }

void PBDSKEL (void)
        {
        int k,i,j,c0,l0;

        CUP(4,1);
        printf ("        6         5         4         3         2
1");
        CUP(8,67);//TextColor(magenta);
        printf("computer");

        for (i=0;i<=1;i++) {
                for (j=1;j<=6;j++)  {
                        c0=basec+(j-1)*incc;
                        l0=basel+i*incl;
                        CUP(l0,c0);
                        CUB(1);
                        for (k=1;k<=incl-1;k++) {
                                printf("|");
                                CUU(1);
                                CUB(1);
                                }
                        printf(".");

                        for (k=1;k<=incc-1;k++)
                                printf("_");

                        }
                }

        for (j=1;j<=6;j++) {
                c0=basec+(j-1)*incc;
                l0=basel+incl;
                printf("\n");
                CUP( l0,c0);
                CUD(1);
                CUB(1);
                printf(".");
                for (k=1;k<=incc-1;k++)
                        printf("_");
                }
```

```
        printf(".");
        for (i=1;i>=0;i--) {
                c0=basec+6*incc;
                l0=basel+i*incl;
                printf("\n");
                CUP(l0,c0);
                for (k= 1;k<=incl-1;k++) {
                        printf("|");
                        CUU(1);
                        CUB(1);
                        }
                printf(".");
                }
        CUP(19,1);

        printf("        1         2         3         4         5
6");
        CUP(14,67);//TextColor(white);
        printf("User");
        }

void MAKE_SOUND (void)
        {
/*        noise=0;
        while noise <25000 do
                begin
                noise=noise+1000;
                sound(noise);delay(5);
                end;
        NoSound; */
        //printf("%c",'\a');
        }

void PLAY (int ci,int s)
        {
        int oc,m,i,c,l;
        i=s;
        c=ci;
        oc=c;//player
        if (board.b[c][i]!=0) {
                do
                { m=board.b[c][i];
                board.b[c][i]=0;
                PHOLE(c,i,board.b[c][i]);
                while (m!=0) {
                        m=m-1;
                        if (c==0)  {
                                i=i-1;
                                if (i==0) {
                                        c=1;
```

```
                                        i=1;
                                        }
                                }
                        else    {
                                i=i+1;
                                if (i==7) {
                                        c=0;
                                        i=6;
                                        }
                                }
                        board.b[c][i]=board.b[c][i]+1;
                        PHOLE(c,i,board.b[c][i]);
                        if ((board.b[c][i]==4) && (m!=0)) {
                                board.w[c]=board.w[c]+4;
                                MAKE_SOUND();
                                PWVAL(board.w[0],board.w[1]);
                                board.b[c][i]=0;
                                PHOLE(c,i,board.b[c][i]);
                                if ((board.w[0]+board.w[1])==44) {
                                        board.w[c]=board.w[c]+4;
                                        MAKE_SOUND();
                                        PWVAL(board.w[0],board.w[1]);
                                        for (l=1;l<=6;l++) {
                                                board.b[0][l]=0;
                                                board.b[1][l]=0;
                                                }
                                        e=true;
                                        }
                                }
                        }
                        if ((board.b[c][i]==4) && (m==0)) {
                                board.w[oc]=board.w[oc]+4;
                                MAKE_SOUND();
                                PWVAL(board.w[0],board.w[1]);
                                board.b[c][i]=0;
                                PHOLE(c,i,board.b[c][i]);
                                if ((board.w[0]+board.w[1])==44) {
                                        board.w[oc]=board.w[oc]+4;
                                        MAKE_SOUND();
                                        PWVAL(board.w[0],board.w[1]);
                                        for (l=1;l<=6;l++) {
                                                board.b[0][l]=0;
                                                board.b[1][l]=0;
                                                }
                                        e=true;
                                        PBDVAL();
                                        }
                                }
                        } while ( !((board.b[c][i]==1) || (board.b[c]
[i]==0)) );
```

```
                }//end if option not 0
        }//end play

struct mvt MOVE (struct bdt board,int ci,int s)
        {
        int oc,m,i,c,l;
        bool e = false;
        struct mvt mvret;
        i=s;
        c=ci;
        oc=c;
        if (board.b[c][i]!=0) {
                do
                {
                m=board.b[c][i];
                board.b[c][i]=0;
                while (m!=0) {
                        m=m-1;
                        if (c==0)  {
                                i=i-1;
                                if (i==0) {
                                        c=1;
                                        i=1;
                                        }
                                }
                        else    {
                                i=i+1;
                                if (i==7) {
                                        c=0;
                                        i=6;
                                        }
                                }
                        board.b[c][i]=board.b[c][i]+1;
                        if ((board.b[c][i]==4) && (m!=0)) {
                                board.w[c]=board.w[c]+4;
                                board.b[c][i]=0;
                                if ((board.w[0]+board.w[1])==44) {
                                        board.w[c]=board.w[c]+4;
                                        for (l=1;l<=6;l++) {
                                                board.b[0][l]=0;
                                                board.b[1][l]=0;
                                                }
                                        e=true;
                                        }
                                }
                        }
                        if ((board.b[c][i]==4) && (m==0)) {
                                board.w[oc]=board.w[oc]+4;
                                board.b[c][i]=0;
                                if ((board.w[0]+board.w[1])==44) {
```

```
                                        board.w[oc]=board.w[oc]+4;
                                        for (l=1;l<=6;l++) {
                                                board.b[0][l]=0;
                                                board.b[1][l]=0;
                                                }
                                        e=true;
                                        }
                                }
                        } while ( !((board.b[c][i]==1) || (board.b[c]
[i]==0)) );
                }
        mvret.board=board;
        mvret.e=e;
        return mvret;
        }

struct evt EVAL (int c,int n,struct bdt board)
        {
        int mval,mii,ci,cval,i,j,l,x,y;
        struct bdt bcopy;
        struct mvt breturn;
        struct evt evalret;
        bool e,reach;

        //run=0; //{evaluation run}
        e=false;
        if (c==0) cval=-1000; else cval=1000;

        bcopy=board;
        i=1;
        ci=i;
        while (i<=6)  {

                x=0;//computer stones
                y=0;//user stones
                reach=false;
                for (l=1;l<=6;l++) {
                        x=x+board.b[0][l];//opponent count
                        y=y+board.b[1][l];//player board count
                        }
                for (l=1;l<=6;l++) {

                        if ( ( (c==0) && (board.b[0][l]>= (7-l))
&&(y==0))
                                || ((c==1) && (board.b[1]
[l]>=l)&&(x==0)))
                        reach=true;
                }

                if( (board.b[c][i]!=0)
```

```
                        && (       ((c==1) && (reach && (board.b[1]
[i]>=(7-1))&& (y==0)))
                                || ((c==1) && (x>0))
                                || ((c==0) && (reach && (board.b[0]
[i]>=i) && (x==0)))
                                || ((c==0) && (y>0))
                                ) )

                        {
                        breturn=MOVE(bcopy,c,i);
                        e=breturn.e;
                        bcopy=breturn.board;

                        if ((n==0) || e) {
                                mval=bcopy.w[0]-bcopy.w[1];
                                }
                        else {evalret=EVAL (1-c,n-1,bcopy);
                                mval=evalret.val;
                                        }

/*                      switch (c) {
                        case 0: if (mval >= cval) {
                                cval=mval;
                                ci=i;
                                }
                        case 1:if (mval <cval)  {
                                cval=mval;
                                ci=i;
                                }
                            }//end switch */

                        if (c==0) {//computer-plus, max
                                if (mval >= cval) {
                                cval=mval;
                                ci=i;
                                }
                        }
                        else {//player-minus, min
                                if (mval <cval)  {
                                cval=mval;
                                ci=i;
                                }
                        }

                        }//end valid move
                   e=false;
                   i=i+1;
                   bcopy=board;
                }//end while for all 6
        evalret.val=cval;
```

```
evalret.ii=ci;
return evalret;
}//end EVAL
```