```cpp
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#include <string.h>

//wata
#include <string>
#include <unistd.h>
#include <iostream>


#include "GameMaster.h"
#include "Board.h"

using namespace std;

#define PLAYERNUM 2
#define BUFFER_SIZE 256
#define DATASIZE 4//ä¸\200åº¦ã\201«ã\202\204ã\202\212ã\201¨ã\202\212ã\201\225ã\202\214ã\202\213ã\203\207ã\203¼ã\202¿ã\201®æ\225°
#define TIME_LIMIT -1
#define BLACKSOCKET 0
#define WHITESOCKET 1

int flagforPS =0;
/* ã\203\235ã\203¼ã\203\210ç\225ªå\217•ã\200\201ã\202½ã\202±ã\203\203ã\203\210 */
char ClientIP[PLAYERNUM][80]={"160.12.172.5","160.12.172.5"};
unsigned short port[2]={9800,9810};
int dstSocket[PLAYERNUM];  // ç\233¸æ\211\213
int status[PLAYERNUM];

/* sockaddr_in æ§\213é\200 ä½\223 */
struct sockaddr_in dstAddr[PLAYERNUM];
int dstAddrSize[PLAYERNUM];
char Teaminfo[20];
char TeamName[PLAYERNUM][40];

/* å\220\204ç¨®ã\203\221ã\203©ã\203¡ã\203¼ã\202¿ */
int numrcv[PLAYERNUM];
char buffer[BUFFER_SIZE];


void kodama(){

  cout<<"ã\202¢ã\202¿ã\203\203ã\202¯ã\203\201ã\203£ã\203¼ã\203³ã\202¹!!"<<endl;
  cout<<"ã\200\200ã\200\200ã\200\200ï¼\217ï¿£ï¿£ï¿£ã\203½"<<endl;
  cout<<"ã\200\200ã\200\200ï¼\217ï¼\217ã\203¾ï½¿))ï¾\230)iï¼¼"<<endl;
  cout<<"ã\200\200 /\"å½¡'ã\203¾ï¾\204ï½¿ï¾\202ã\203\216ã\203\237ï½\234"<<endl;
  cout<<"ã\200\200ï½\234å½¡ã\200\200 _ ï¼¿ã\200\200 ï¾\220ï½\234"<<endl;
  cout<<"ã\200\200 ï¾\232å½¡ ï¼¿_ã\200\200 ï¼¿ã\203½/"<<endl;
  cout<<"ã\200\200(6ï½²ã\200\200ã\202£oã\200\201 ï½¨oï½¤â\210¥"<<endl;
  cout<<"ã\200\200ã\203½|ã\200\200ï½¤ï¼¿ï¾\211 iï½¤_ ï½\234"<<endl;
  cout<<"ã\200\200 ï½\234ï¼¿ã\200\200rï½¤_ï½\202ã\203½/"<<endl;
  cout<<"ï¼¿ï¼\217rä¸\211ï¾\220Dã\203¼-â\206\222ã\202½"<<endl;
  cout<<"ã\200\200|ï½\234^ rj ï¼¿\"ã\203\216-ã\200\201"<<endl;
  cout<<"ã\200\200'ï¾\211ã\200\200 )ã\203½_/|ï¼¼ã\200\200ã\203½"<<endl;
  cout<<"ã\200\200â\210§ _â\210§ /ã\203½|ã\200\200ã\203½ |"<<endl;
  //sleep(3);
}

void colombia(string answer){


  cout<<"ã\200\200 ã\200\200 ã\200\200 ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200 ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200 ã\200\200ã\200\200ã\200\200 ,.ã\201¸"<<endl;
  cout<<"ã\200\200ã\200\200___ ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200 ã\200\200 ã\200\200 ã\200\200 ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200 ã\200\200ã\200\200ã\200\200ã\200\200ï¾\221ã\200\200ã\200\200i"<<endl;
  cout<<"ã\200\214 ï¾\213_iã\200\211ã\200\200ã\200\200ã\200\200 ã\200\200 ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200 ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200 ã\202\235ã\200\200ã\200\210"<<endl;
  cout<<"ï¾\204ã\200\200ã\203\216 ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200 ã\200\200ã\200\200ã\200\200iã\203\213(()"<<endl;
  cout<<"i ã\200\200{ ã\200\200 ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200 ã\200\200ã\200\200ã\200\200ï¼¿ï¼¿ï¼¿_ ã\200\200 ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200| ã\200\200ã\203½"<<endl;
  cout<<"iã\200\200ã\200\200iã\200\200ã\200\200ã\200\200 ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200 ã\200\200ï¼\217__,ã\200\200 , â\200\220-ï¼¼  ã\200\200  ã\200\200 ã\200\200i  ã\200\200}"<<endl;
  cout<<"|ã\200\200ã\200\200 iã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200 ã\200\200ã\200\200ï¼\217ï¼\210â\227\217) ã\200\200 ( â\227\217 ) ï¼¼ã\200\200ã\200\200ã\200\200   {ï½¤ã\200\200 Î»"<<endl;
  cout<<"ã\200\200ã\203\210ï¼\215â\224¤.ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200  ã\200\200ï¼\217 ã\200\200 ï¼\210__äºº__ï¼\211  ã\200\200ï¼¼ã\200\200ã\200\200 ,ã\203\216ã\200\200ï¿£,!"<<endl;
  cout<<"ã\200\200iã\200\200ã\200\200ã\200\200ã\202\235ï½¤_ ã\200\200ã\200\200ã\200\200|ã\200\200ã\200\200ã\200\200  Â´ï¿£' ã\200\200 ã\200\200 ã\200\200|ã\200\200,. 'Â´ï¾\212ã\200\200,!"<<endl;
  cout<<".ã\200\200ã\203½ã\200\201 ã\200\200ã\200\200ã\200\200ï½\200'ã\200\200ï½¤,__ï¼¼ ã\200\200ã\200\200 ã\200\200 ã\200\200ã\200\200ã\200\200ã\200\200 ï¼\217\"ã\200\200ï¼¼ ã\200\200ï¼\217"<<endl;
  cout<<"ã\200\200ã\200\200ã\200\200ï¼¼ã\203\216ã\200\200ï¾\211ã\200\200ã\200\200ã\200\200ï¾\212ï¿£r/:::râ\200\225--â\200\225/::ï¼\227ã\200\200ã\200\200 ï¾\211ã\200\200ã\200\200   ï¼\217"<<endl;
  cout<<"ã\200\200 ã\200\200ã\200\200 ã\203½_ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ã\203½::ã\200\210'::. :'|::/       /ã\200\200ã\200\200ã\200\200,."<<endl;
  cout<<"ã\200\200         'ï½° ï½¤ã\200\200     ï¼¼ã\203½::. ;::ï¼\232|/ã\200\200  ã\200\200ï½\222'\""<<endl;
  cout<<"ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ï¼\217ï¿£äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214ã\203½"<<endl;
  cout<<"          | ç-\224 |           "<<answer<<"                 â\224\202|"<<endl;
  cout<<"ã\200\200ã\200\200ã\200\200ã\200\200ã\200\200ï¼¼ï¼¿äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214äº\214ã\203\216"<<endl;
  cout<<""<<endl;
  cout<<""<<endl;
  //sleep(3);
}
```

```
 90:
 91: ostream& operator<<(ostream& os, const Point& p)
 92: {
 93:         string s = p;
 94:         os << s;
 95:         return os;
 96: }
 97:
 98: void assign_stringtochar(string input, char *output){
 99:   for(int i=0;i<DATASIZE;i++){
100:     output[i]=input[i];
101:   }
102: }
103:
104: void assign_chartostring(char *input, string &output){
105:   for(int i=0;i<DATASIZE;i++){
106:     output[i]=input[i];
107:   }
108: }
109:
110: void assign_char(char *input, char *output){
111:   for(int i=0;i<DATASIZE;i++){
112:     output[i]=input[i];
113:   }
114: }
115:
116:
117: void GFprocess(ConsoleBoard &board){
118:   if(board.countDisc(BLACK)>board.countDisc(WHITE)){
119:     cout<<"WINER is BLACK"<<endl;
120:   }
121:   else if(board.countDisc(BLACK)<board.countDisc(WHITE)){
122:     cout<<"WINER is WHITE"<<endl;
123:   }
124:   else{
125:     cout<<"DRAW"<<endl;
126:   }
127:   cout << "----------------Game Finish----------------" << endl;
128:   //*********************é\200\232ä¿¡ã\203\235ã\202¤ã\203³ã\203\210*********
*********
129:   char send_data[DATASIZE];
130:   send_data[0]='0';
131:   send_data[1]='0';
132:   send_data[2]='0';
133:   send_data[3]=GFFLAG;
134:
135:   send(dstSocket[0], send_data,strlen(send_data)+1,0);
136:   send(dstSocket[1], send_data,strlen(send_data)+1,0);
137: }
138:
139: void time_out(Color current_color){
140:
141:   string in;
142:
143:   if(current_color == BLACK){
144:     cout  << "time out" <<endl;
145:     cout << "LOSER is BLACK "<<endl;
146:     cout << "please input any character and push Enter key"<<endl;
147:     cin>>in;
148:   }
149:   else{
150:     cout  << " time out" <<endl;
151:     cout << "LOSER is WHITE"<<endl;
152:     cout << "please input any character and push Enter key"<<endl;
153:     cin>>in;
154:   }
155:   //*********************é\200\232ä¿¡ã\203\235ã\202¤ã\203³ã\203\210*********
*********
156:   //client ã\201¯å\213\235æ\211\213ã\201«çµ\202äº\206?
157:   char send_data[DATASIZE];
158:   send_data[0]='0';
159:   send_data[1]='0';
160:   send_data[2]='0';
161:   send_data[3]=GFFLAG;
162:
163:   send(dstSocket[0], send_data,strlen(send_data)+1,0);
164:   send(dstSocket[1], send_data,strlen(send_data)+1,0);
165:   exit(0);
166: }
167:
168: void against_rules(Color current_color){
169:
170:   string in;
171:
172:   if(current_color == BLACK){
173:     cout  << "you against the rule" <<endl;
174:     cout << "LOSER is BLACK "<<endl;
175:     cout << "please input any character and push Enter key"<<endl;
176:     cin>>in;
177:   }
178:   else{
179:     cout  << "you against the rule" <<endl;
180:     cout << "LOSER is WHITE"<<endl;
181:     cout << "please input any character and push Enter key"<<endl;
182:     cin>>in;
183:   }
184:   //*********************é\200\232ä¿¡ã\203\235ã\202¤ã\203³ã\203\210*********
********
185:   //client ã\201¯å\213\235æ\211\213ã\201«çµ\202äº\206?
186:   char send_data[DATASIZE];
187:   send_data[0]='0';
188:   send_data[1]='0';
189:   send_data[2]='0';
190:   send_data[3]=GFFLAG;
191:
192:   send(dstSocket[BLACKSOCKET], send_data,strlen(send_data)+1,0);
193:   send(dstSocket[WHITESOCKET], send_data,strlen(send_data)+1,0);
194:   exit(0);
195: }
196:
197: bool attack_chance(int current_color, ConsoleBoard &board, string premove){
198:
199:   if(current_color == BLACK){
200:     cout << "So, Black, What number?"<<endl;
201:
202:   }
203:   else{
204:     cout << "So, White, What number?"<<endl;
205:   }
206:
207:
208:   string in(DATASIZE,0);
209:   char buffer[BUFFER_SIZE];
210:   char send_data[DATASIZE];
```

```
  211:
  212:
  213:
  214:   //**********************é\200\232ä¿¡ã\203\235ã\202¤ã\203³ã\203\210**********
*********
  215:
  216:   premove[3] = ACFLAG;
  217:   if(flagforPS ==true){
  218:     premove[3]= premove[3] | PSFLAG;
  219:   }
  220:   assign_stringtochar(premove, send_data);
  221:   if(current_color==BLACK){
  222:     if(TIME_LIMIT == send(dstSocket[BLACKSOCKET], send_data,strlen(send_data)+
1,0)) time_out(current_color);
  223:     if(TIME_LIMIT == recv(dstSocket[BLACKSOCKET], buffer, BUFFER_SIZE, 0)) tim
e_out(current_color);
  224:   }else{
  225:      if(TIME_LIMIT == send(dstSocket[WHITESOCKET], send_data,strlen(send_data)
+1,0)) time_out(current_color);
  226:      if(TIME_LIMIT == recv(dstSocket[WHITESOCKET], buffer, BUFFER_SIZE, 0)) ti
me_out(current_color);
  227:   }
  228:   assign_chartostring(buffer, in);
  229:   cout<<"revese Disk:"<<in<<endl;
  230:   //*************************************************
  231:
  232:
  233:   //debug
  234:   //cout<<"char to string buffer0"<<buffer<<endl;
  235:   //cout<<"char to string buffer1"<<buffer<<endl;
  236:   //cout<<"char to string buffer2"<<buffer<<endl;
  237:   //cout<<"char to string buffer3"<<hex<<(int)buffer<<endl;
  238:
  239:   Point p(in);
  240:
  241:   //ã\202\202ã\201\227æ\214\207å®\232ã\201¾ã\201\231ã\201«ç\233¸æ\211\213ã\201
®ç\237³ã\201\214ã\201ªã\201\213ã\201£ã\201\237ã\202\211
  242:   if(board.getColor(p) != (-current_color)){
  243:     cout<<"Against the rules"<<endl;
  244:     against_rules(current_color);
  245:     }
  246:   else{//ç\233¸æ\211\213ã\201®ç\237³ã\201\214ã\201\202ã\201£ã\201\237ã\202\211
  247:     colombia(in);
  248:     board.Reverse_disk(p, current_color);
  249:     assign_char(buffer, send_data);
  250:     send_data[3]=ARFLAG;
  251:     //**********************é\200\232ä¿¡ã\203\235ã\202¤ã\203³ã\203\210********
***********
  252:     if(current_color==BLACK){
  253:       if(TIME_LIMIT == send(dstSocket[WHITESOCKET], send_data,strlen(send_data
)+1,0)) time_out(current_color);
  254:     }else{
  255:       if(TIME_LIMIT == send(dstSocket[BLACKSOCKET], send_data,strlen(send_data
)+1,0)) time_out(current_color);
  256:     }
  257:     //*****************************************************************
  258:   }
  259:   if(board.isGameOver()){
  260:     board.print();
  261:     cout << "Black Disk:" << board.countDisc(BLACK) << " ";
  262:     cout << "White Disk:" << board.countDisc(WHITE) << " ";
  263:     cout << "Empty:" << board.countDisc(EMPTY) << endl;
  264:     GFprocess(board);
  265:   }
  266:   return true;
  267: }
  268:
  269:
  270:
  271:
  272:
  273: int
  274: main() {
  275:   for(int i=0;i<PLAYERNUM;i++){
  276:     dstAddrSize[i]  = sizeof(dstAddr);
  277:   }
  278:
  279:   /*************************************************************/
  280:   /* ç\233¸æ\211\213å\205\210ã\202¢ã\203\211ã\203¬ã\202¹ã\201®å\205¥å\212\233
*/
  281:
  282:
  283:   cout<<"BLACK IP address :";
  284:   cin>>ClientIP[BLACKSOCKET];
  285:
  286:   cout<<"BLACK port :";
  287:   cin>>port[BLACKSOCKET];
  288:
  289:   cout<<"WHITE IP adrress :";
  290:   cin>>ClientIP[WHITESOCKET];
  291:
  292:   cout<<"WHITE port :";
  293:   cin>>port[WHITESOCKET];
  294:
  295:   /*************************************************************/
  296:   for(int i=0;i<PLAYERNUM;i++){
  297:     /* sockaddr_in æ§\213é\200 ä½\223ã\201®ã\202»ã\203\203ã\203\210 */
  298:     memset(&dstAddr[i], 0, sizeof(dstAddr));
  299:     dstAddr[i].sin_port = htons(port[i]);
  300:     dstAddr[i].sin_family = AF_INET;
  301:     dstAddr[i].sin_addr.s_addr = htonl(INADDR_ANY);
  302:
  303:     /* ã\202½ã\202±ã\203\203ã\203\210ã\201®ç\224\237æ\210\220 */
  304:     dstSocket[i] = socket(AF_INET, SOCK_STREAM, 0);
  305:
  306:     /* ã\202½ã\202±ã\203\203ã\203\210ã\201®ã\203\220ã\202¤ã\203³ã\203\211 */
  307:     bind(dstSocket[i], (struct sockaddr *) &dstAddr[i], sizeof(dstAddr[i]));
  308:
  309:     /* æ\216¥ç¶\232ã\201®è¨±å\217¯ */
  310:     listen(dstSocket[i], 1);
  311:
  312:     /* æ\216¥ç¶\232ã\201®å\217\227ä»\230ã\201\221 */
  313:     printf("Waiting for connection ...\n");
  314:     dstSocket[i] = accept(dstSocket[i], (struct sockaddr *) &dstAddr[i], &dstA
ddrSize[i]);
  315:     printf("Connected from %s\n", inet_ntoa(dstAddr[i].sin_addr));
  316:
  317:     /* ã\203\221ã\202±ã\203\203ã\203\210å\217\227ä¿¡ */
  318:     numrcv[i] = recv(dstSocket[i], buffer, BUFFER_SIZE, 0);
  319:     cout<<"received:"<<buffer<<endl;
  320:
  321:     if(i==0){
  322:       strcpy(Teaminfo,"Black");
  323:       //å\220\215å\211\215ç\231»é\214²å\207¦ç\220\206
```

```
  324:       strcpy(TeamName[i],buffer);
  325:       cout<<"Black's Team name : "<<TeamName[i]<<endl;
  326:     }else {
  327:       strcpy(Teaminfo,"White");
  328:       //å\220\215å\211\215ç\231»é\214²å\207¦ç\220\206
  329:       strcpy(TeamName[i],buffer);
  330:       cout<<"White's Team name : "<<TeamName[i]<<endl;
  331:     }
  332:
  333:     send(dstSocket[i], Teaminfo,strlen(Teaminfo)+1, 0);
  334:   }
  335:
  336:
  337:   /*ã\202²ã\203¼ã\203 ã\202¹ã\202¿ã\203¼ã\203\210*/
  338:   ConsoleBoard board;
  339:   //watanabe wrote 2017/3/31
  340:   string premove(DATASIZE,0);
  341:   premove[0]='0';
  342:   premove[1]='0';
  343:   premove[2]='0';
  344:   premove[3] = 0;
  345:   int attack_cahnce_status=false;
  346:   int mt_status_BLACK=false;
  347:   int mt_status_WHITE=false;
  348:
  349:   //å\210¶é\231\220æ\231\202é\226\223
  350:   struct timeval limit_tv;
  351:   limit_tv.tv_sec  = 1;//sec
  352:   limit_tv.tv_usec = 0;//usec
  353:
  354:   //ã\202¼ã\202±ã\203\203ã\203\210ã\201®ã\202¿ã\202¤ã\203 ã\202¢ã\202¦ã\203
\210è¨-å®\232
  355:   for(int i=0;i<PLAYERNUM;i++){
  356:     setsockopt(dstSocket[i], SOL_SOCKET, SO_SNDTIMEO, (char *)&limit_tv, sizeo
f(limit_tv));
  357:   }
  358:
  359:   while(true){
  360:
  361:       board.print();
  362:       cout << "Black Disk:" << board.countDisc(BLACK) << " ";
  363:       cout << "White Disk:" << board.countDisc(WHITE) << " ";
  364:       cout << "Empty:" << board.countDisc(EMPTY) << endl;
  365:
  366:       int current_color =  board.getCurrentColor();
  367:       if(current_color == BLACK){
  368:         cout<<"Black Turn("<<TeamName[BLACKSOCKET]<<")"<<endl;
  369:       }
  370:       else{
  371:         cout<<"White Turn("<<TeamName[WHITESOCKET]<<")"<<endl;
  372:       }
  373:
  374:
  375:       cout << endl << endl;
  376:
  377:       //ç\211¹æ®\212ã\203«ã\203¼ã\203«ã\201®è¿½å\212  attack cance
  378:       //æ®\213ã\202\212ã\201¾ã\201\231ã\201\21410ä»¥ä¸ \213 ã\201\213ã\201¤ 20ç
\237³ä»¥ä¸ \212è² ã\201\221ã\201¦ã\201\204ã\202\213 ã\202¢ã\202¿ã\203\203ã\202¯ã\203
\201ã\203£ã\203³ã\202¹ã\201\214èµ•ã\201\223ã\201£ã\201¦ã\201\204ã\201ªã\201\204
  379:       if( (board.countDisc(EMPTY) <= 10) && ( (board.countDisc(-current_color)
-board.countDisc(current_color) ) >= ATTACKNUM) && (attack_cahnce_status == false ) ){
  380:         kodama();
  381:         attack_cahnce_status = true;
  382:         attack_chance(current_color, board, premove);
  383:
  384:         board.print();
  385:         cout << "Black Disk:" << board.countDisc(BLACK) << " ";
  386:         cout << "White Disk:" << board.countDisc(WHITE) << " ";
  387:         cout << "Empty:" << board.countDisc(EMPTY) << endl;
  388:
  389:         if(current_color == BLACK){
  390:           cout<<"Black Turn"<<endl;
  391:         }
  392:         else{
  393:           cout<<"White Turn"<<endl;
  394:         }
  395:         premove[0]='0';
  396:         premove[1]='0';
  397:         premove[2]='0';
  398:         }
  399:
  400:         cout << "input your move: ";
  401:         Point p;
  402:
  403:         string in(DATASIZE,0);
  404:
  405:         //**********************é\200\232ä¿¡ã\203\235ã\202¤ã\203³ã\203\210****
**************
  406:         //send data to player and recieve data from player
  407:         premove[3] = 0;
  408:         if(flagforPS == true)  premove[3] = premove[3] | PSFLAG;
  409:         char send_data[DATASIZE];
  410:         assign_stringtochar(premove, send_data);
  411:         if(current_color == BLACK){
  412:           if(TIME_LIMIT == send(dstSocket[BLACKSOCKET], send_data,strlen(send_
data)+1,0)) time_out(current_color);
  413:           if(TIME_LIMIT == recv(dstSocket[BLACKSOCKET], buffer, BUFFER_SIZE,0)
) time_out(current_color);
  414:         }else {
  415:           if(TIME_LIMIT ==  send(dstSocket[WHITESOCKET], send_data,strlen(send
_data)+1,0)) time_out(current_color);
  416:           if(TIME_LIMIT ==  recv(dstSocket[WHITESOCKET], buffer, BUFFER_SIZE,
0)) time_out(current_color);
  417:         }
  418:         flagforPS  = false;
  419:         assign_chartostring(buffer, in);
  420:         cout<<in<<endl;
  421:         cout<<"in0"<<in[0]<<endl;
  422:         cout<<"in1"<<in[1]<<endl;
  423:         cout<<"in2"<<in[2]<<endl;
  424:         cout<<"in3"<<hex<<in[3]<<endl;
  425:       /*********************************************************/
  426:
  427:
  428:         //æ\211\213å\210¤æ\226-
  429:         //ã\203\221ã\202¹ã\201ªã\202\211
  430:       if(in[0] == 'p')
  431:         {
  432:           // ã\203\221ã\202¹
  433:           if(!board.pass()){
  434:             cerr << "you can't pass " << endl;
  435:             against_rules(current_color);
  436:           }else {
  437:             cout<<"PASS"<<endl;
```

```
438:             flagforPS= true;
439:             premove[0]='0';
440:             premove[1]='0';
441:             premove[2]='0';
442:           }
443:           if(board.isGameOver())
444:             { board.print();
445:               cout << "Black Disk:" << board.countDisc(BLACK) << " ";
446:               cout << "White Disk:" << board.countDisc(WHITE) << " ";
447:               cout << "Empty:" << board.countDisc(EMPTY) << endl;
448:               GFprocess(board);
449:               break;
450:             }
451:
452:           continue;
453:         }
454:
455:       //point ã\201«å¤\211æ\217\233
456:       try
457:         {
458:           Point parse(in);
459:           p = parse;
460:         }
461:
462:       //é\226\223é\201\225ã\201£ã\201\237å\205¥å\212\233
463:       catch(invalid_argument e)
464:         {
465:           cerr <<"wrong your input" << endl;
466:           against_rules(current_color);
467:           continue;
468:         }
469:
470:       //ã\201\212ã\201\221ã\201ªã\201\204å ´æ\211\200
471:       if(board.move(p) == false)
472:         {
473:           cerr << "you can't move the place" << endl;
474:           against_rules(current_color);
475:           continue;
476:         }
477:
478:       premove = in;
479:
480:
481:       //test mtflag
482:       if((p.flag & MTFLAG) == MTFLAG){
483:         if( (current_color == WHITE) && (mt_status_WHITE == true)){//ã\202\202
ã\201\227MTFLAGã\201\214å\220\214ã\201\230ã\203\227ã\203¬ã\202¤ã\203¤ã\203¼ã\201§äº
\214åº¦ã\201\202ã\201£ã\201\237ã\202\211é\201\225å\217\215 flagã\201«ã\202\210ã\202
\212currentcolorã\201¯ã\201\235ã\201®ã\201¾ã\201¾ã\201§ã\202\210ã\201\204
484:           against_rules(current_color);
485:         }
486:         else if( (current_color == BLACK) && (mt_status_BLACK == true)){
487:           against_rules(current_color);
488:         }
489:         else if(board.getTurns()<10){
490:           against_rules(current_color);
491:         }
492:         cout<<"Still my turn!!!!!!!!!"<<endl;
493:
494:         if(current_color==WHITE){
495:           mt_status_WHITE=true;
496:         }
497:         else{
498:           mt_status_BLACK=true;
499:         }
500:         premove[3]= MTFLAG;
501:         //mtã\201\213ã\201¤ps
502:         if(flagforPS == true){
503:           premove[3]=MTFLAG | PSFLAG;
504:         }
505:
506:         //**********************é\200\232ä¿¡ã\203\235ã\202¤ã\203³ã\203\210****
**************
507:         char send_data[DATASIZE];
508:         assign_stringtochar(premove, send_data);
509:         if(current_color==BLACK){
510:           if(TIME_LIMIT == send(dstSocket[WHITESOCKET], send_data,strlen(send_
data)+1,0)) time_out(current_color);
511:         }else{
512:           if(TIME_LIMIT == send(dstSocket[BLACKSOCKET], send_data,strlen(send_
data)+1,0)) time_out(current_color);
513:         }
514:         premove[0]='0';
515:         premove[1]='0';
516:         premove[2]='0';
517:         premove[3]=0;
518:         continue;
519:       }
520:
521:
522:       if(board.isGameOver())
523:         {
524:           board.print();
525:           cout << "Black Disk:" << board.countDisc(BLACK) << " ";
526:           cout << "White Disk:" << board.countDisc(WHITE) << " ";
527:           cout << "Empty:" << board.countDisc(EMPTY) << endl;
528:           GFprocess(board);
529:           break;
530:         }
531:     }
532:
533: }
534:
```