# Project 1

## <Liar Dice>

## <Version 2.0>

Name: Haolan Ye (Benjamin)

Class: CSC-17A 43950

Date: 06/04/2015

# Contents

# 1. Introduction( http://en.wikipedia.org/wiki/Liar's_dice ):

Five dice are used per player with dice cups used for concealment. Each round, each player rolls a "hand" of dice under their cup and looks at their hand while keeping it concealed from the other players.

The first player begins bidding, announcing any face value and the number of dice that the player believes are showing that value, under all of the cups in the game. Ones are often wild, always counting as the face of the current bid. Each player has two choices during their turn: to make a higher bid, or challenge the previous bid - typically with a call of "liar." Raising the bid means either increasing the quantity, or the face value, or both.

If the current player challenges the previous bid, all dice are revealed. If the bid is valid (at least as many of the face value and any wild aces are showing as were bid), the bidder wins. Otherwise, the challenger wins.

That game I made was created based on the way I played in China. Because of difference between different regions, some rules of Liar Dice are different. In China, there are some addition rules (also in my game):

- From the beginning, ones are wild unless you bid 3 ones
- After bidding ones, ones cannot be wild anymore
- You could bid only 3 fives, which means you bid there are 3 fives but not including ones
- After bidding only 3 fives, ones cannot be wild anymore
- The number of face of first bid has to be greater than 1.5*players

# 2. Summary:

| Total Line of Code | 1000+ |
|---|---|
| Comment Line | - |
| Variable | - |
| Function | - |

This game contains most concepts that we have learned in the class. I used pointer with player (structure) and used structure to record the dices that each player has. In the structure of player, there is also a tag (integer) for the player. I will use the tag when someone wants to challenge. The game will write the data of players into binary file, and after someone challenge, it read the file to a new players array. Afterward, it will get the result by using the new players array.

# 3. Problems during coding

a) **Limit the player input with correct format**

When player doesn't challenge, he need to make a higher bid. Player needs to input a string for bidding. "4 5" means that player bids 4 fives. "4n5" means that player bids 4 fives only (ones cannot be wild at that time).

b) **Get the playing order for the players**

At the beginning of the game, it will randomly get the playing order for the players. In the rest of the game, players bid and challenge based on that order. I used a switch statement in a do-while statement to randomly access. It loops until someone challenges.

**c) What should AI do?**

There is no a specific algorithm for the AI in that game. I made the AI based on what I think when I play Liar Dice. There are lots of possibilities that happens when AI determine challenging or not.

- When AI doesn't have the dices that bided by previous players
- When AI only have one that bided by previous players

… etc.

When AI needs to bid higher, AI should sometimes lie and sometimes tell the truth. Therefore, I set the possibility that AI lie to 2/5. When AI tells the truth, he will bid based on what dices he has. When AI lies, he will randomly select one face of dice that does not exist in his dices.

**d) One is wild**

Mostly, ones are wild unless you bid "3 fives only" or "3 ones" (Both are example). Therefore, I need a Boolean to record one is wild or not. After one is not wild, the number of each dices doesn't count ones.

# 4.Pseudo Code

Set seed for random number

Sign in/sign up (input name, password, email)

Introduce the game

Display menu (play, add coins, or exit)

When Play, Prompt players for the number of players

Roll the dices for players

Initialize based on the number of player

Display player's dices

Randomly choose a player be the first bidder

Begin biding until someone challenge {

  Someone bid first (based on random select)

  Other players determine challenge or not

  Bid in players order

}

Show dices of all players

Display the result of the game {

  Check the number of face bided is greater than the number of dices that players have exactly

}

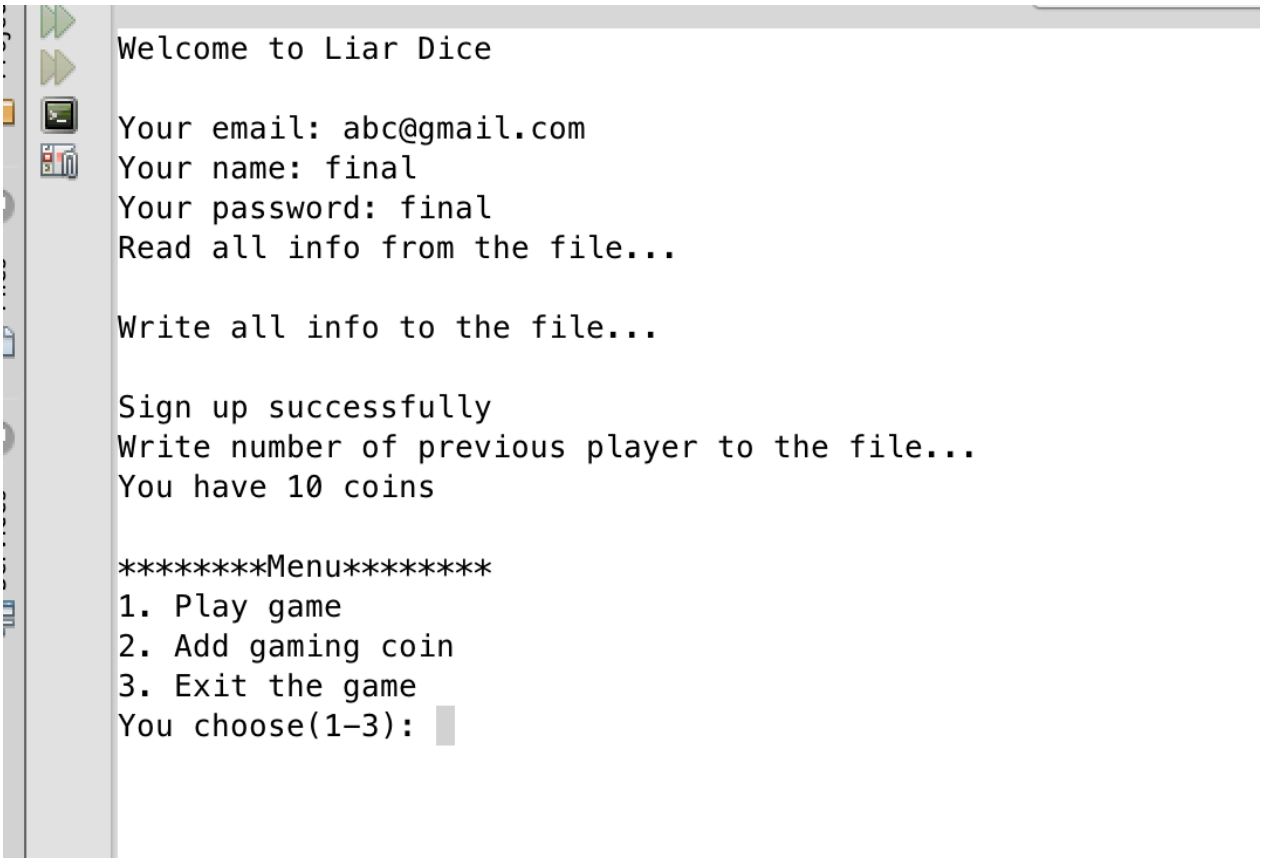
update the information of players with binary file

When add coins, prompt user input the card number

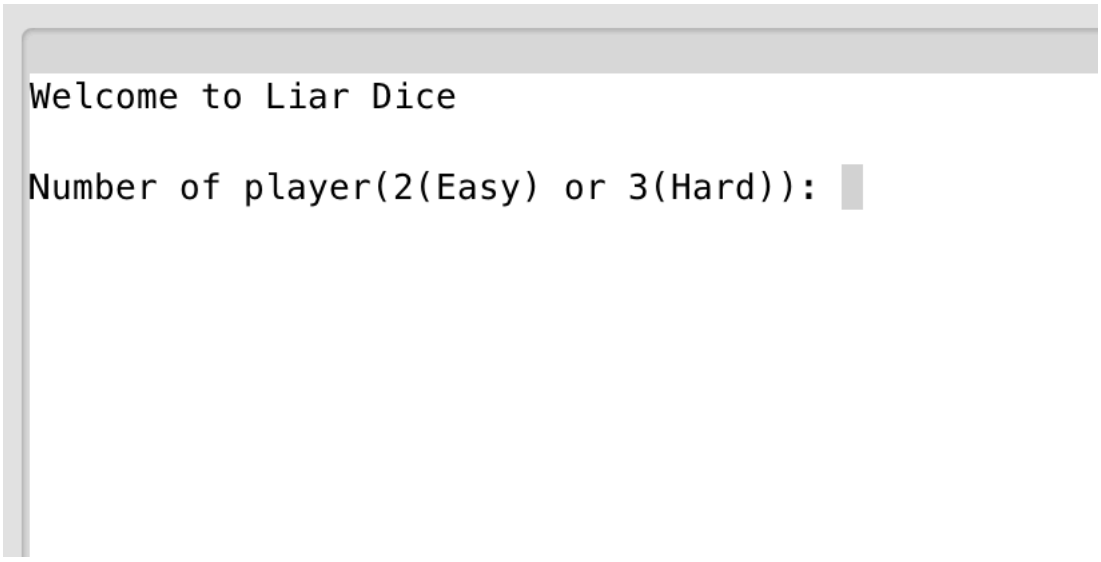
Deallocate memory

# 5. Screen Shot

1) Ask for number of player




2) Ask for number of player



3) Randomly choose a player to be the first bid, then you can challenge

```
Welcome to Liar Dice

Number of player(2(Easy) or 3(Hard)): 3
Write to the file...

Your    dice: 6 5 5 3 5
AI #1 bid 4  4s
Would you like to challenge?(Y or N):
```

4) If challenge, the result will come up

```
Welcome to Liar Dice

Number of player(2(Easy) or 3(Hard)): 3
Write to the file...

Your    dice: 6 5 5 3 5
AI #1 bid 4  4s
Would you like to challenge?(Y or N): y
You Challenge
Read from the file...


Your    dice: 6 5 5 3 5
AI #1's dice: 4 4 4 5 5
AI #2's dice: 5 5 5 3 4

Totally, there are 4 4s
Your challenge failed
```

5) If not challenge, your turn to bid

```
Welcome to Liar Dice

Number of player(2(Easy) or 3(Hard)): 3
Write to the file...

Your    dice: 4 1 3 1 5
AI #2 bid 4  3s
Would you like to challenge?(Y or N): n
AI #1 does not challenge
Your bidding: format:"3 4"(means u bid 3 4s,and 1s are wild) or "4n5"
Your bidding:
```

6) After you bid, if AI(s) doesn't challenge, it's their turn to bid, and ask you challenge or not

```
Welcome to Liar Dice

Number of player(2(Easy) or 3(Hard)): 3
Write to the file...

Your    dice: 4 1 3 1 5
AI #2 bid 4  3s
Would you like to challenge?(Y or N): n
AI #1 does not challenge
Your bidding: format:"3 4"(means u bid 3 4s,and 1
Your bidding: 4 5
You bid 4  5s
AI #1 does not challenge
AI #2 does not challenge
AI #1 bid 4  6s
Would you like to challenge?(Y or N):
```

7) Add gaming coins

```
********Menu********
1. Play game
2. Add gaming coin
3. Exit the game
You choose(1-3): 2
How many gaming coin you would like to buy($1 for 1 coin): 100
Your credit/debit card number: 4916141313641562
You bought 100 gaming coins successfully
Read all info from the file...
Update 115 to the file
Write all info to the file...

Now, you have 115 gaming coins
Click Enter to continue...
```

# 6. Libraries

## *a*. System libraries

- ✓ **<iostream>**
- ✓ **<cstdlib>**
- ✓ **<ctime>**
- ✓ **<string>**
- ✓ **<vector>**
- ✓ **<fstream>**

## *b*. System libraries

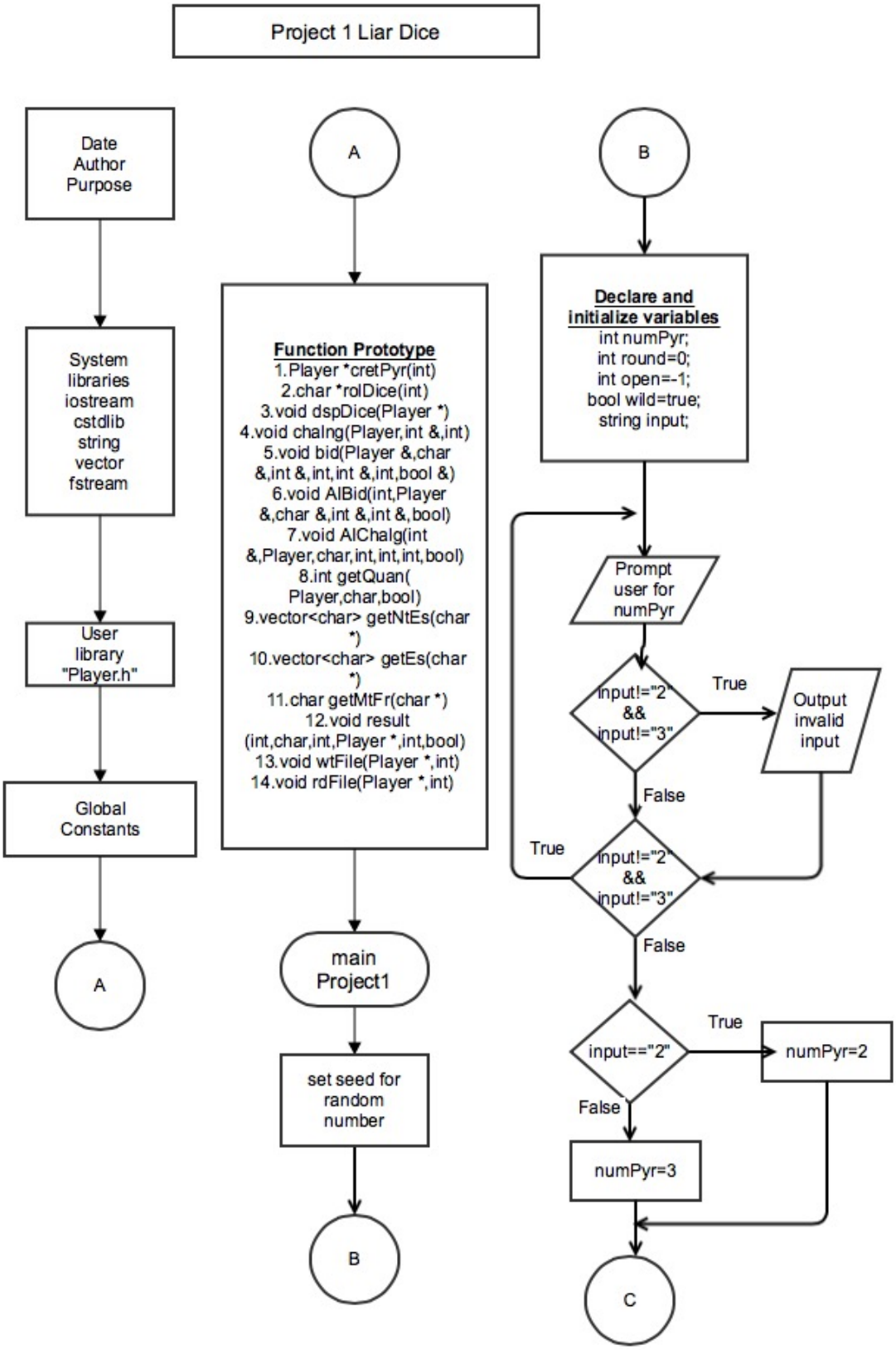
- ✓ **AI.h**
- ✓ **Player.h**

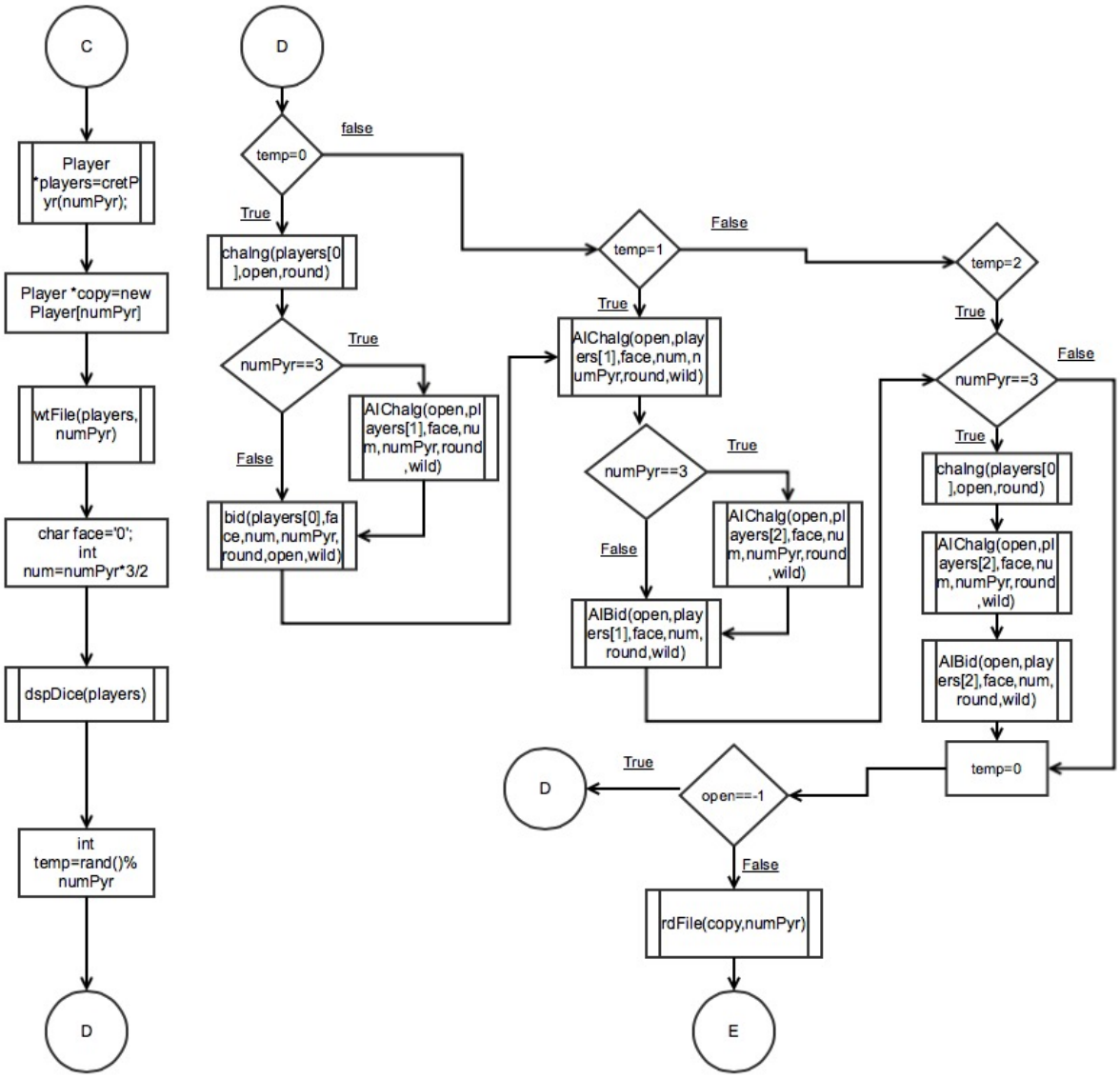
# 7. Concept covered

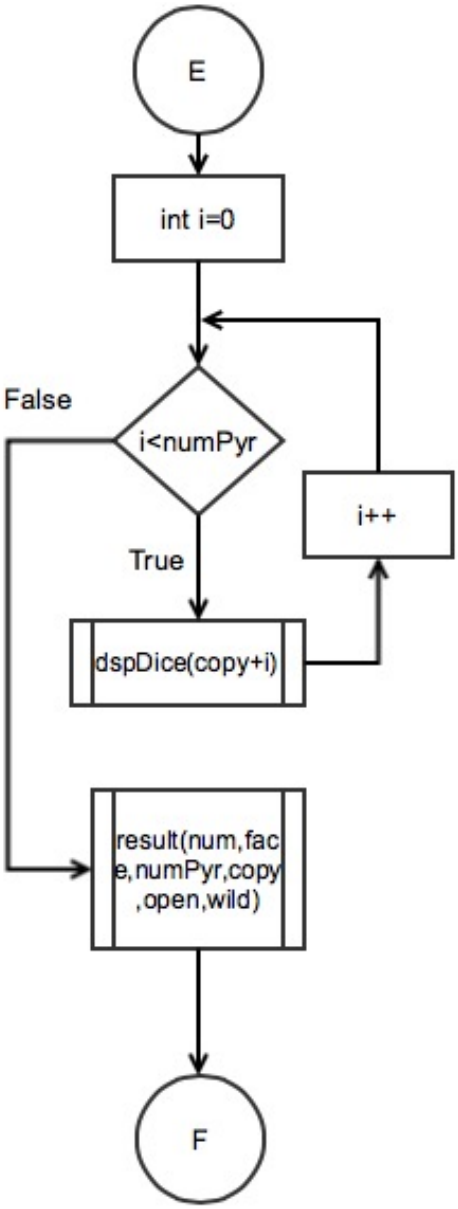
| Concept | Type | Code | Location(line) |
|---|---|---|---|
| Pointer with structure | Info * | Info *infor=new Info[getNInf()]; | 259 in player.cpp |
| Type casting | static_cast<type > | static_cast<unsigned short>(time(0)) | 35 in main |
| Binary file | fstream output | fstream out; | 274 in player.cpp |
| | fstream input | fstream in; | 260 in player.cpp |
| string | string | string ans="N" | 161 |
| class | class | class Player | 14 in player.h |
| getter | get | int  getOrdr() const {return order;} | 64 in player.h |
| setter | set | void setNInf(int); | 33 in player.h |
| inline function | | void setOrdr(int n) {order=n;}; | 51 player.h |
| Constructor | | Player::Player() | 29 player.cpp |
| Overload function | | void bid() | 25 in AI.h |
| abstract base class | | virtual void bid()=0; | AbsPlayer.h |

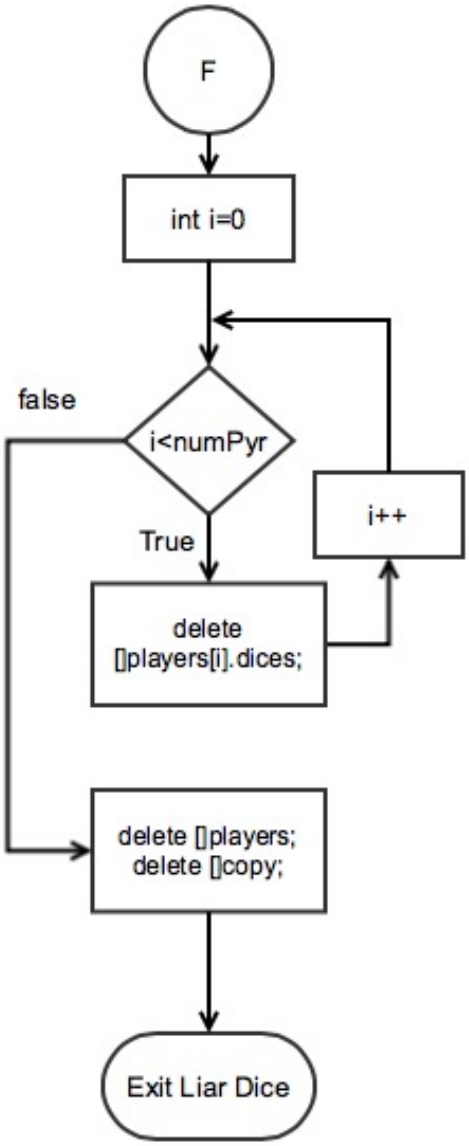
| | | | |
|---|---|---|---|
| vector | vector<char> | vector<char> nExist | 15 AI.h |
| Sorting | Sorting | for(int i=0;i<5;i++) {<br>for(int j=i+1;j<6;j++) {<br>if(nExist[i]<nExist[j]) {<br>char temp=nExist[i];<br>nExist[i]=nExist[j];<br>nExist[j]=temp;<br>}<br>}<br>} | 332 |
| Array of Object | | AI *a=new AI[np-1]; | 60 in main |
| Instance variables | static int | static int open | 21 in Player.h |
| Static Member Functions | | static void setNumC(); | 23 in Player.h |
| Operator Overloading | | T &operator[](const int &); | 32 in aVector.h |
| inheritance | | class AI:public Player | 12 in Player.cpp |
| Template class | | template <class T><br>class aVector | 16 in aVector.h |
| Exception | | try {<br>aptr = new T[usdSize];<br>} catch (bad_alloc) {<br>memError();<br>} | 45 in aVector.h |

# 8. Flowchart in Project 1 (1) Main flowchart (3 pages)

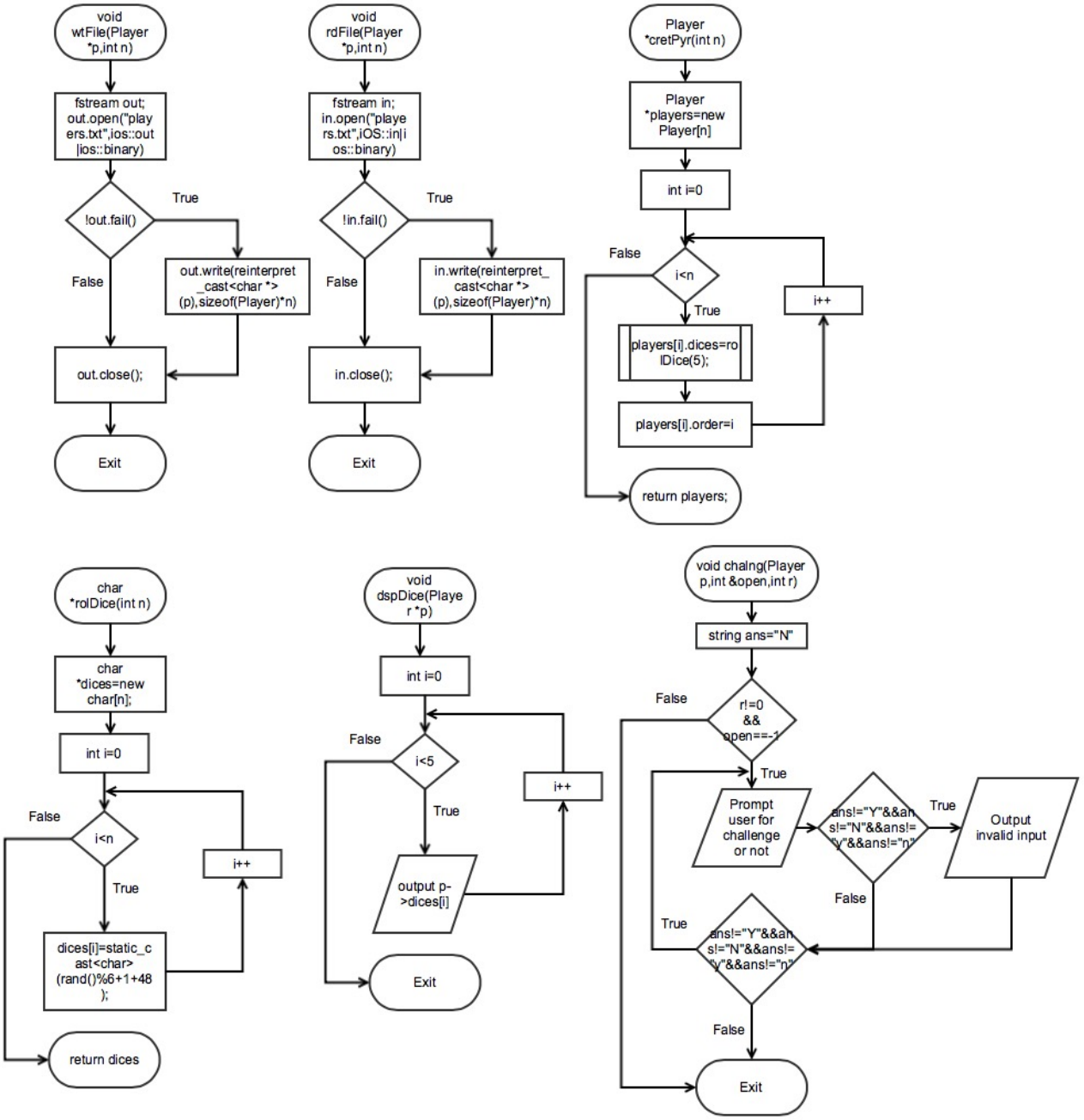


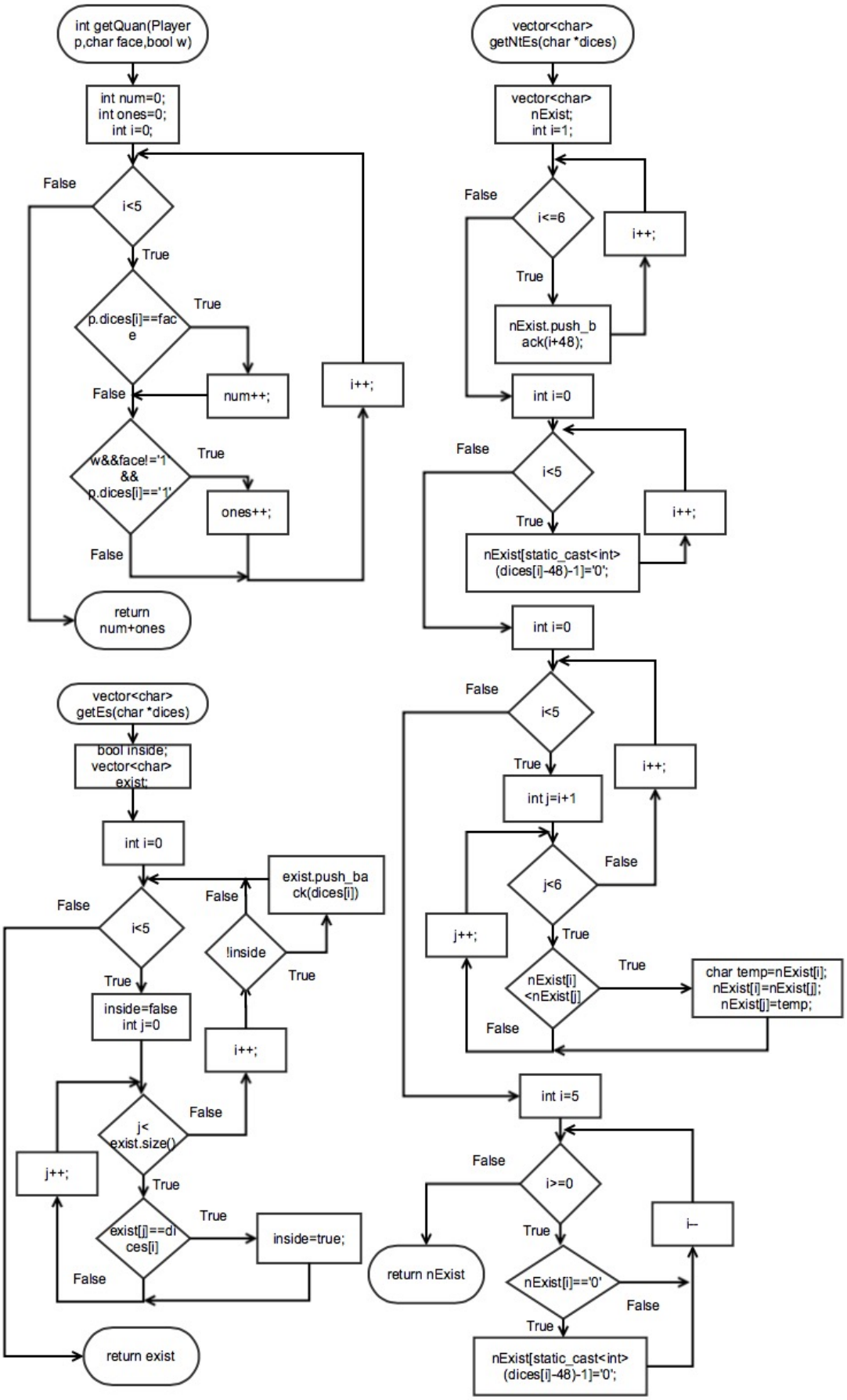

C
Player *players=cretPyr(numPyr);
Player *copy=new Player[numPyr]
wtFile(players, numPyr)
char face='0'; int num=numPyr*3/2
dspDice(players)
int temp=rand()% numPyr
D
D
temp=0
false
True
chalng(players[0],open,round)
numPyr==3
True
AIChalg(open,players[1],face,num,numPyr,round,wild)
False
bid(players[0],face,num,numPyr,round,open,wild)
temp=1
False
True
AIChalg(open,players[1],face,num,numPyr,round,wild)
numPyr==3
True
AIChalg(open,players[2],face,num,numPyr,round,wild)
False
AIBid(open,players[1],face,num,round,wild)
temp=2
True
numPyr==3
False
True
chalng(players[0],open,round)
AIChalg(open,players[2],face,num,numPyr,round,wild)
AIBid(open,players[2],face,num,round,wild)
temp=0
D
True
open==-1
False
rdFile(copy,numPyr)
E

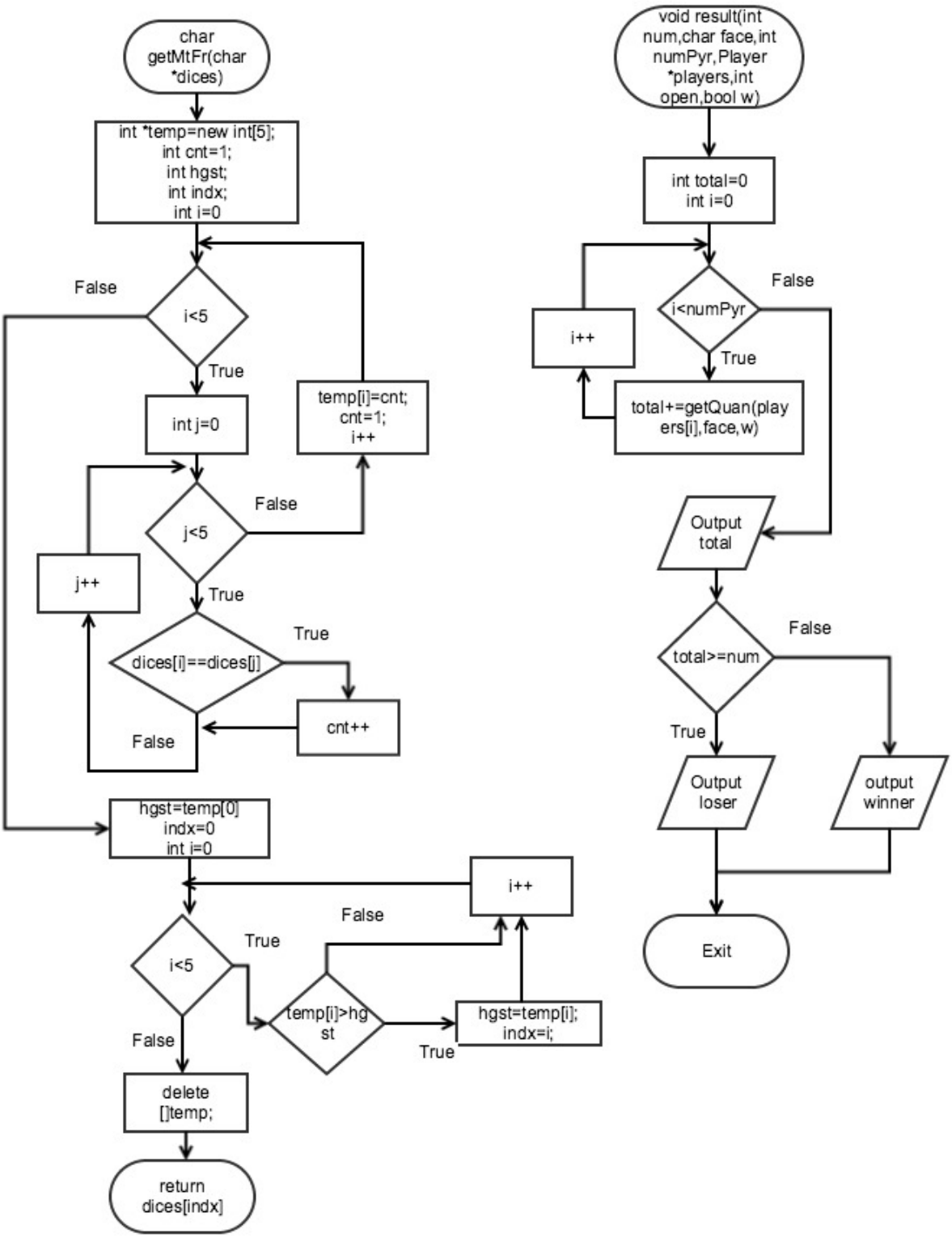

E
int i=0
False
i<numPyr
True
i++
dspDice(copy+i)
result(num,face,numPyr,copy,open,wild)
F

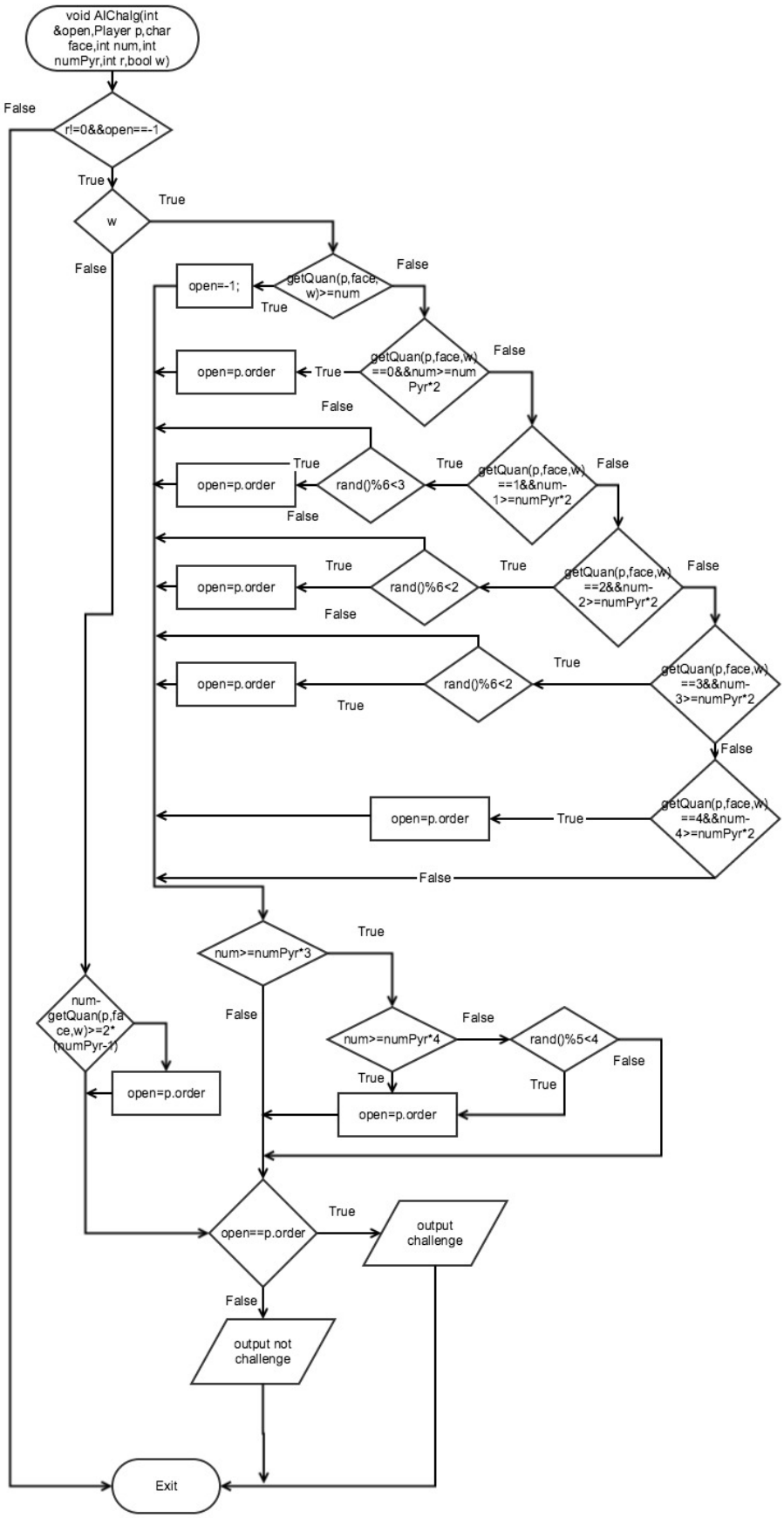

F
int i=0
false
i<numPyr
True
i++
delete []players[i].dices;
delete []players;
delete []copy;
Exit Liar Dice

# (2) Function flowchart (3 pages)

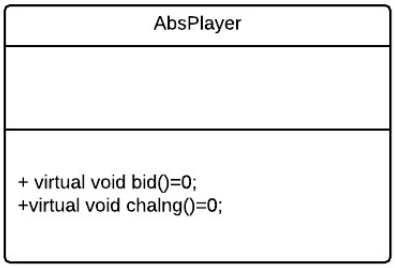


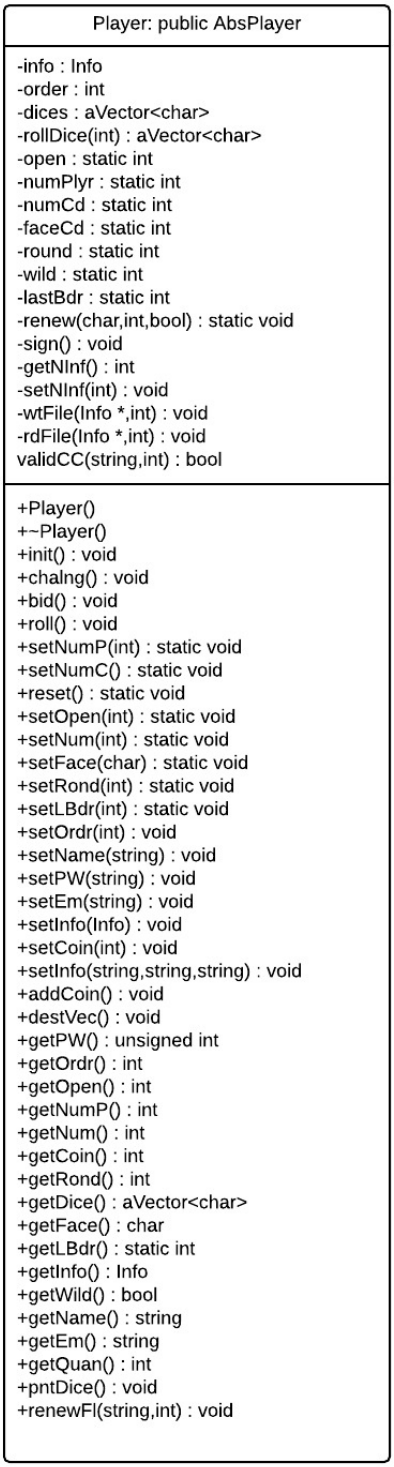

int getQuan(Player p,char face,bool w)
int num=0;
int ones=0;
int i=0;
i<5
False
True
p.dices[i]==face
True
num++;
False
w&&face!='1' && p.dices[i]=='1'
True
ones++;
False
i++;
return num+ones
vector<char> getNtEs(char *dices)
vector<char> nExist;
int i=1;
i<=6
False
True
nExist.push_back(i+48);
i++;
int i=0
i<5
False
True
nExist[static_cast<int>(dices[i]-48)-1]='0';
i++;
int i=0
i<5
False
True
int j=i+1
j<6
False
True
nExist[i]<nExist[j]
True
char temp=nExist[i];
nExist[i]=nExist[j];
nExist[j]=temp;
False
j++;
i++;
int i=5
i>=0
False
True
nExist[i]=='0'
True
False
nExist[static_cast<int>(dices[i]-48)-1]='0';
i--
return nExist
vector<char> getEs(char *dices)
bool inside;
vector<char> exist;
int i=0
i<5
False
True
inside=false
int j=0
j< exist.size()
False
True
exist[j]==dices[i]
True
inside=true;
False
j++;
!inside
False
True
exist.push_back(dices[i])
i++;
return exist

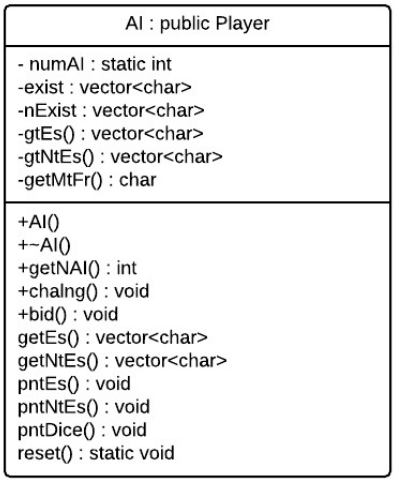

char getMtFr(char *dices)
int *temp=new int[5];
int cnt=1;
int hgst;
int indx;
int i=0
i<5
False
True
int j=0
temp[i]=cnt;
cnt=1;
i++
j<5
False
True
j++
dices[i]==dices[j]
True
False
cnt++
hgst=temp[0]
indx=0
int i=0
i++
i<5
True
False
temp[i]>hgst
True
hgst=temp[i];
indx=i;
False
delete []temp;
return dices[indx]
void result(int num,char face,int numPyr,Player *players,int open,bool w)
int total=0
int i=0
i<numPyr
False
True
i++
total+=getQuan(players[i],face,w)
Output total
total>=num
False
True
Output loser
output winner
Exit


void AIChalg(int &open,Player p,char face,int num,int numPyr,int r,bool w)
r!=0&&open==-1
False
True
w
True
False
getQuan(p,face,w)>=num
False
True
open=-1;
getQuan(p,face,w)==0&&num>=numPyr*2
True
False
open=p.order
getQuan(p,face,w)==1&&num-1>=numPyr*2
True
False
rand()%6<3
True
False
open=p.order
getQuan(p,face,w)==2&&num-2>=numPyr*2
True
False
rand()%6<2
True
False
open=p.order
getQuan(p,face,w)==3&&num-3>=numPyr*2
True
False
rand()%6<2
True
False
open=p.order
getQuan(p,face,w)==4&&num-4>=numPyr*2
True
False
open=p.order
num>=numPyr*3
True
False
num-getQuan(p,face,w)>=2*(numPyr-1)
True
False
open=p.order
num>=numPyr*4
True
False
rand()%5<4
True
False
open=p.order
open==p.order
True
output challenge
False
output not challenge
Exit

# 9. UML


AbsPlayer
+ virtual void bid()=0;
+virtual void chalng()=0;


Player: public AbsPlayer
-info : Info
-order : int
-dices : aVector<char>
-rollDice(int) : aVector<char>
-open : static int
-numPlyr : static int
-numCd : static int
-faceCd : static int
-round : static int
-wild : static int
-lastBdr : static int
-renew(char,int,bool) : static void
-sign() : void
-getNInf() : int
-setNInf(int) : void
-wtFile(Info *,int) : void
-rdFile(Info *,int) : void
validCC(string,int) : bool
+Player()
+~Player()
+init() : void
+chalng() : void
+bid() : void
+roll() : void
+setNumP(int) : static void
+setNumC() : static void
+reset() : static void
+setOpen(int) : static void
+setNum(int) : static void
+setFace(char) : static void
+setRond(int) : static void
+setLBdr(int) : static void
+setOrdr(int) : void
+setName(string) : void
+setPW(string) : void
+setEm(string) : void
+setInfo(Info) : void
+setCoin(int) : void
+setInfo(string,string,string) : void
+addCoin() : void
+destVec() : void
+getPW() : unsigned int
+getOrdr() : int
+getOpen() : int
+getNumP() : int
+getNum() : int
+getCoin() : int
+getRond() : int
+getDice() : aVector<char>
+getFace() : char
+getLBdr() : static int
+getInfo() : Info
+getWild() : bool
+getName() : string
+getEm() : string
+getQuan() : int
+pntDice() : void
+renewFl(string,int) : void


AI : public Player
- numAI : static int
-exist : vector<char>
-nExist : vector<char>
-gtEs() : vector<char>
-gtNtEs() : vector<char>
-getMtFr() : char
+AI()
+~AI()
+getNAI() : int
+chalng() : void
+bid() : void
getEs() : vector<char>
getNtEs() : vector<char>
pntEs() : void
pntNtEs() : void
pntDice() : void
reset() : static void