Project 1

<Liar Dice>

<Version 1.0>

Name: Haolan Ye (Benjamin)

Class: CSC-17A 43950

Date: 04/26/2015

**Contents**

1. **Introduction…………………………………………...2**
2. **Summary…………………………………………........3**
3. **Problems during coding………………………………3**
4. **Pseudo Code………………………………………...…5**
5. **Screen shot…………..………………..…………..........6**
6. **System Libraries…………………………………...….9**
7. **Variable List……………………..…………..………..9**
8. **Function List...………………………………..……...11**
9. **Concept Covered…..………….……..……..……......12**
10. **Flowchart...………………..………………………..14**
11. **Code...……………………..………………………..22**
12. **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

1. **Summary:**

|  |  |
| --- | --- |
| Total Line of Code | 406 |
| Comment Line | 45 |
| Variable | 26 |
| Function | 14 |

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.

1. **Problems during coding**
2. **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).

1. **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.

1. **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.

1. **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.

1. **Pseudo Code**

Set seed for random number

Introduce the game

Prompt players for the number of players

Roll the dices for players

Write the data of players into binary file

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

}

Read the binary file and save the data to a new array of structure

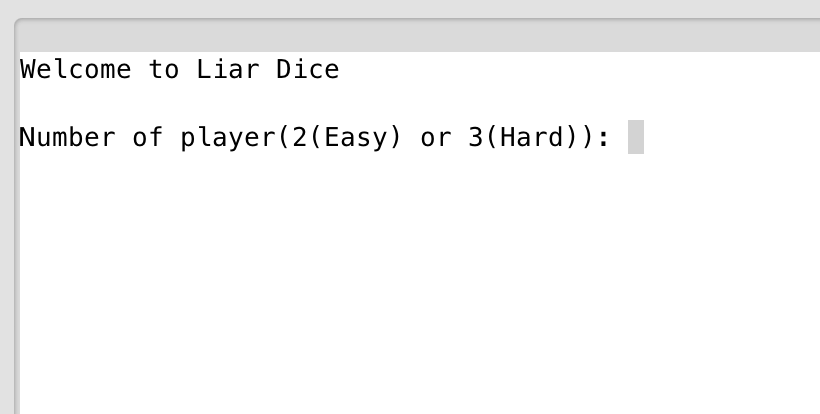
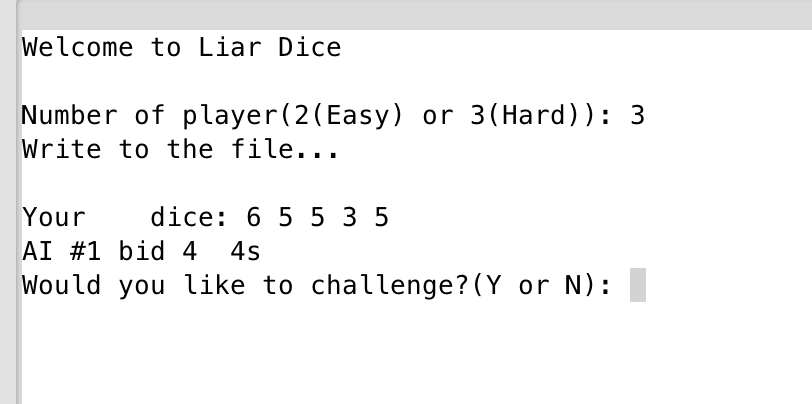
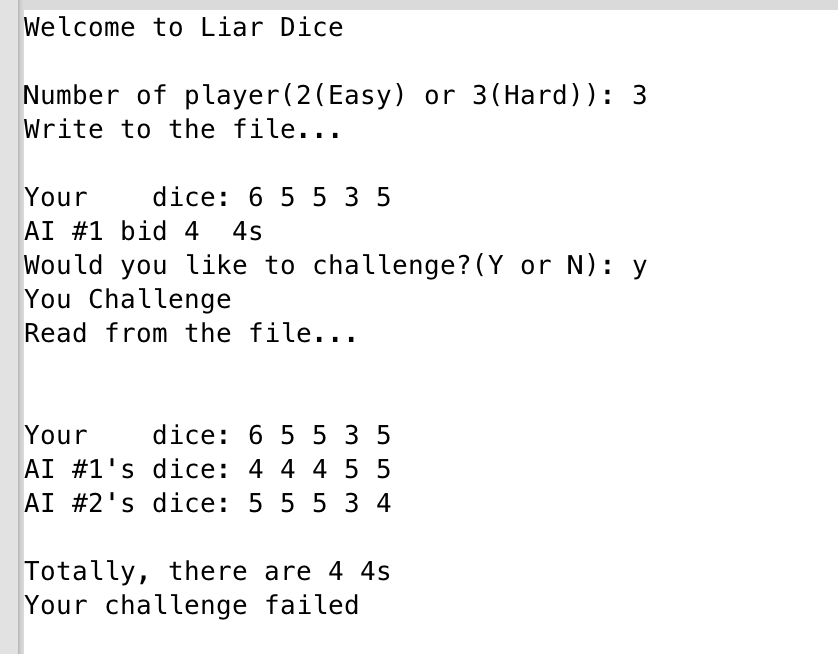
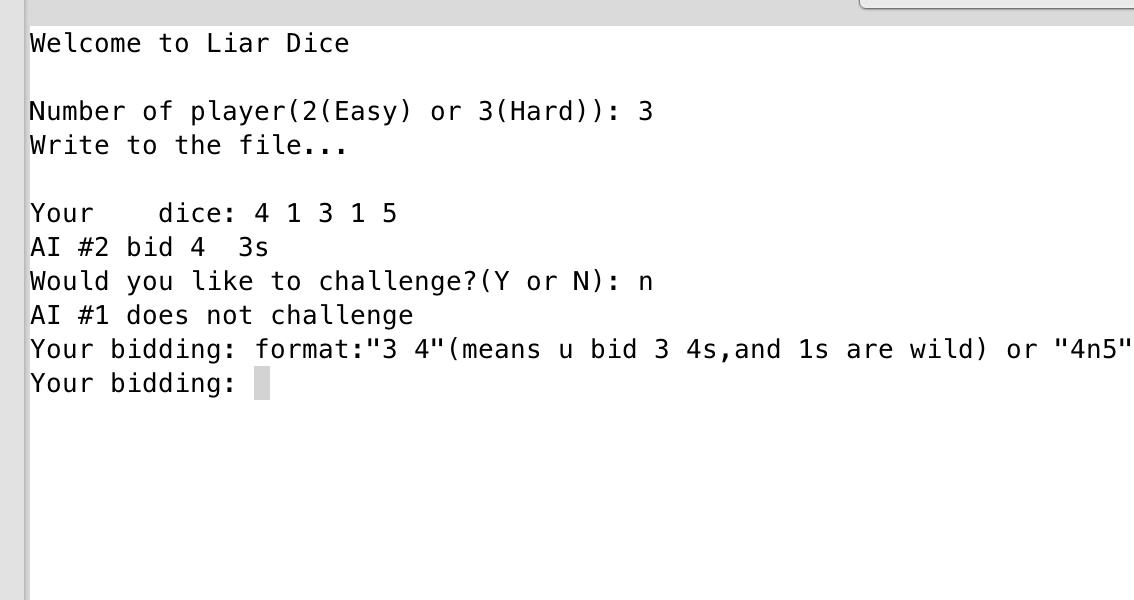
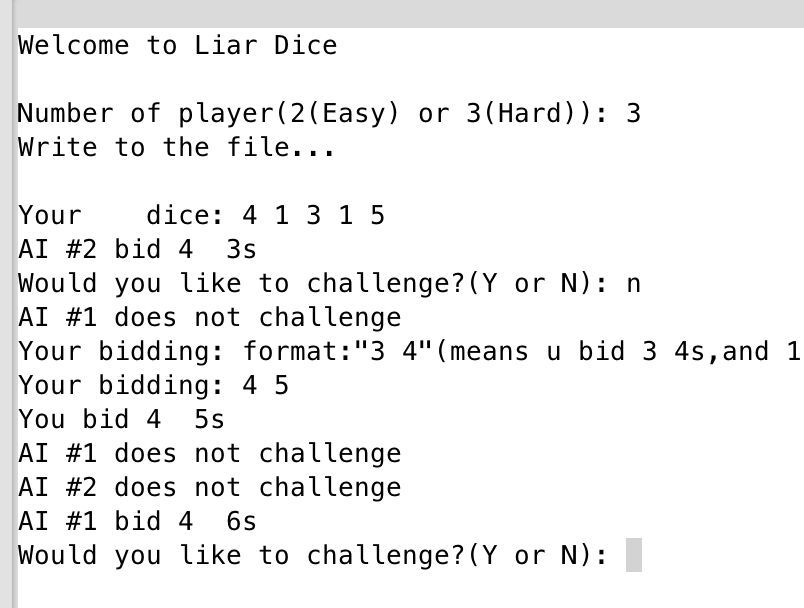
Show dices of all players

Display the result of the game {

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

}

Deallocate memory

1. **Screen Shot**
2. Ask for number of player
3. Randomly choose a player to be the first bid, then you can challenge
4. If challenge, the result will come up
5. If not challenge, your turn to bid
6. After you bid, if AI(s) doesn’t challenge, it’s their turn to bid, and ask you challenge or not
7. **System Libraries**

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

1. **Variables List**

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Variable Name | Description | Declare Location(line) |
| fstream | out | Output to the file | 109 |
|  | in | Read from the file | 119 |
| int | numPyr | Number of player | 42 |
|  | round | How many round | 43 |
|  | open | A tag for open player | 44 |
|  | temp | Temp for randomly get a player | 65 |
|  | numTemp | Number of face bided | 180 |
|  | num | Number of one face of dice | 310 |
|  | ones | How many ones in dices | 311 |
|  | cnt | Count for one face of dice | 366 |
|  | hgst | Highest frequency in dices | 367 |
|  | indx | Index of highest frequency | 368 |
|  | total | Total number of one faces in three players’ hand | 392 |
| int \* | temp | Record the frequency | 365 |
| string | input | Let player to input then check | 46 |
|  | ans | Answer of challenge or not | 161 |
|  | bid | What player bid | 179 |
| Player \* | players | Pointer for players | 57 |
|  | copy | Pointer for players from file | 58 |
| char | fceTemp | Face of dice bided | 181 |
| char \* | dices | Dices for one player | 141 |
| char | guess | The number player guess | 62 |
| vector<char> | nExist | Faces not exist in dices | 322 |
|  | exist | Face exist in dices | 351 |
| boolean | wild | Ones are wild or not | 45 |
|  | invalid | Input invalid or not | 182 |

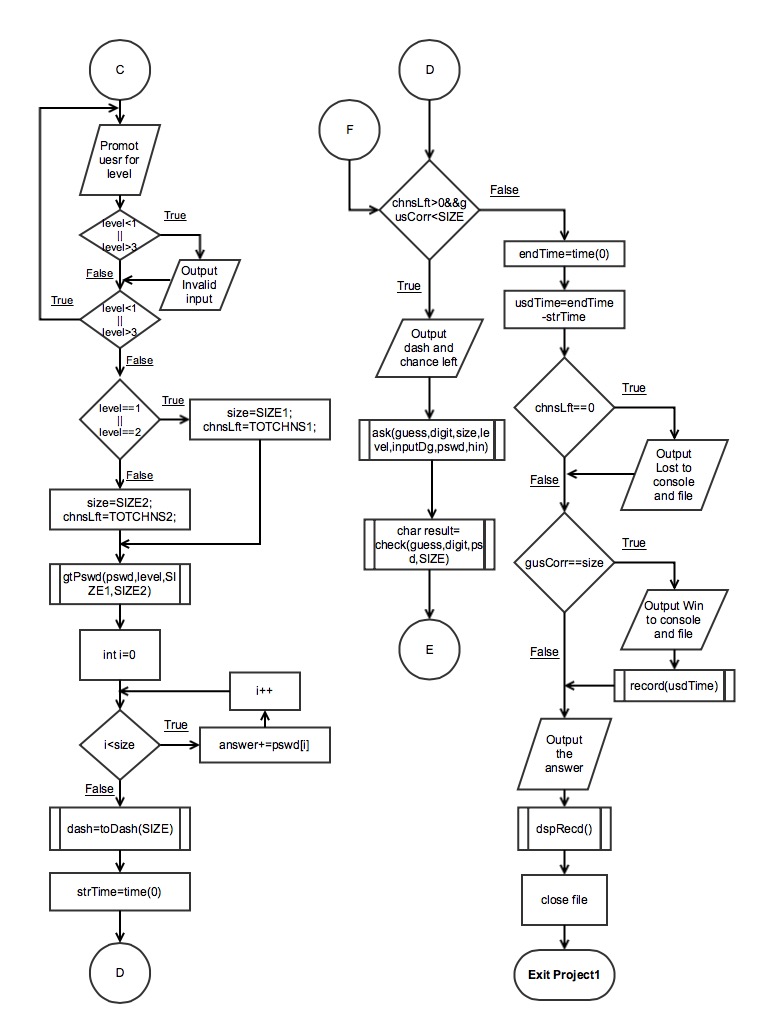
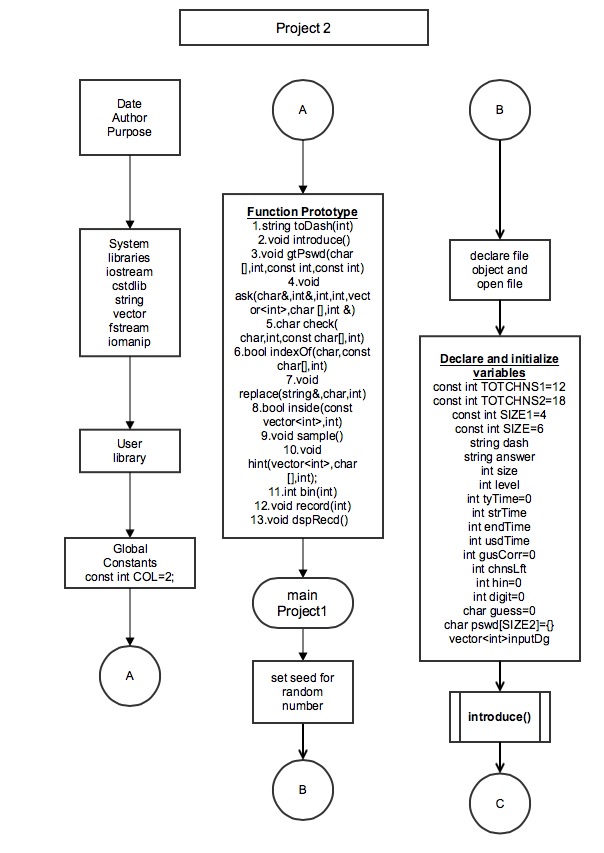
1. **Function List**

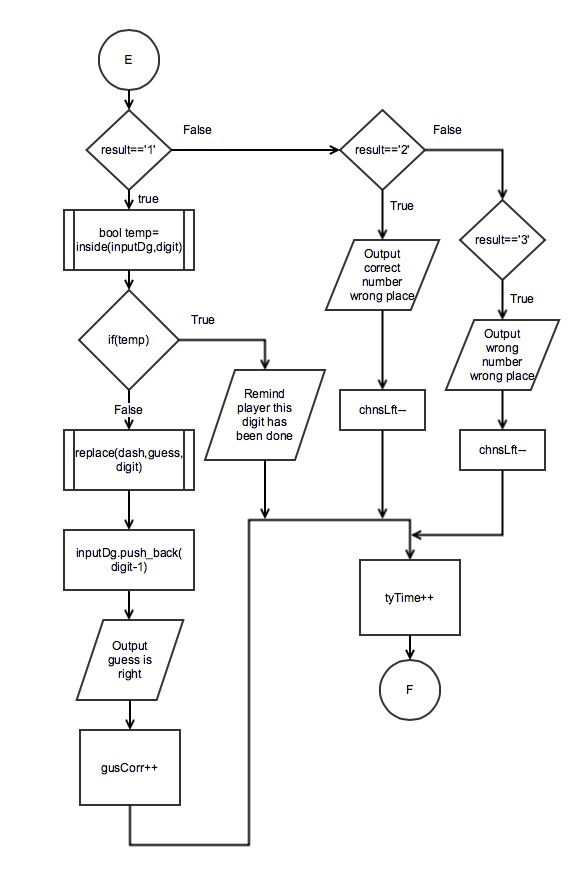
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Name | Argument | Function | Location |
| Player \* | cretPyr | int | create player and roll dice | 129 |
| char \* | rolDice | int | roll 5 dices | 139 |
| void | dspDice | Player \* | display dice of a player | 150 |
| void | chalng | Player,int &,int | Player challenge | 160 |
| void | bid | Player &,char &,int &,int,int &,int,bool & | Player bid | 178 |
| void | AIBid | int,Player &,char &,int &,int &,bool | AI bid | 266 |
| void | AIChalg | int &,Player,char,int,int,int,bool | AI challenge or not | 237 |
| int | getQuan | Player,char,bool | Get the quantity of that face of dice in one AI's hand | 309 |
| vector<char> | getNtEs | char \* | Get the dices that not exist one AI's hand | 321 |
| vector<char> | getEs | char \* | Get the dices that exist one AI's hand | 349 |
| char | getMtFr | char \* | get the most frequent face of dices in one AI's hand | 364 |
| void | result | int,char,int,Player \*,int,bool | Determine who win and lost | 391 |
| void | wtFile | Player \*,int | write the array of Player into file | 108 |
| void | rdFile | Player \*,int | Read the file | 118 |

1. **Concept covered**

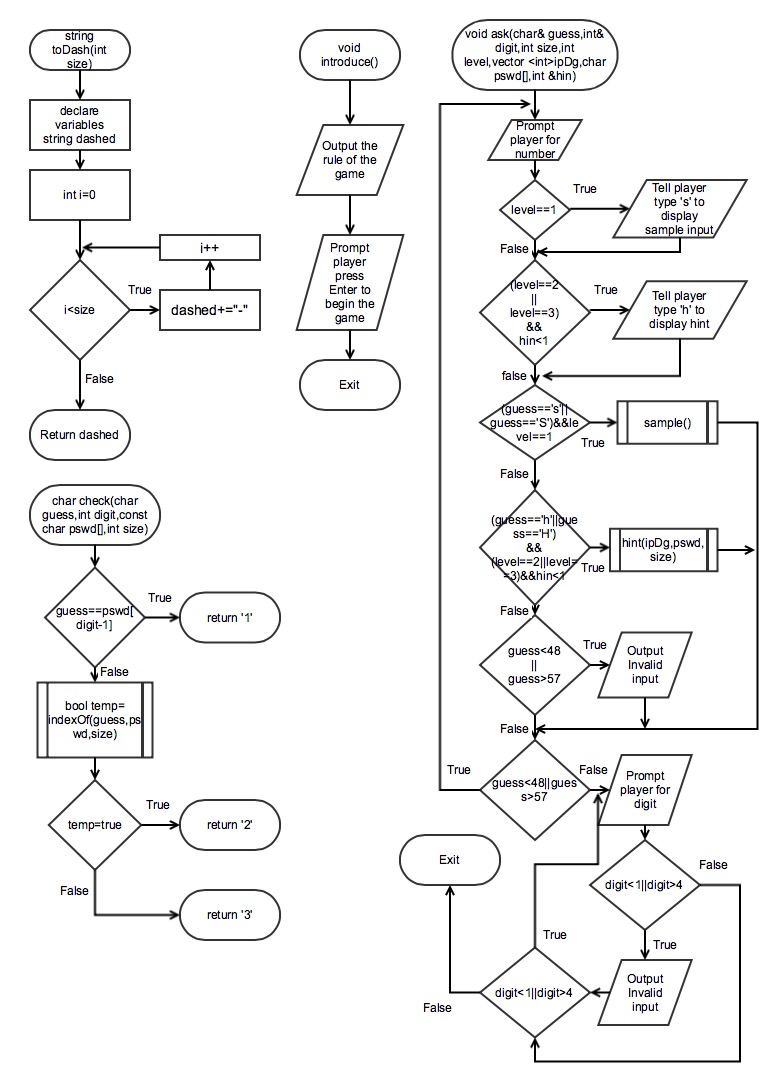
|  |  |  |  |
| --- | --- | --- | --- |
| Concept | Type | Code | Location(line) |
| Pointer with structure | Player \* | Player \*players=cretPyr(numPyr); | 57 |
| Function return pointer |  | Player \*cretPyr(int); | 23 |
| Functionwith structure |  | Player \*cretPyr(int); | 23 |
| Point with array | int \* | int \*temp=new int[5]; | 365 |
| Type casting | static\_cast<type> | static\_cast<unsigned short>(time(0)) | 41 |
| Binary file | fstream output | fstream out; | 109 |
|  | fstream input | fstream in; | 119 |
| string | string | string ans="N" | 161 |
| Switch | switch | switch(temp) | 68 |
| Loop | for | for(int i=1;i<=6;i++) | 324 |
|  | do-while | do {} while(open==-1) | 67 |
| Function | void, int, string, char, bool | string toDash(int) | 23 |

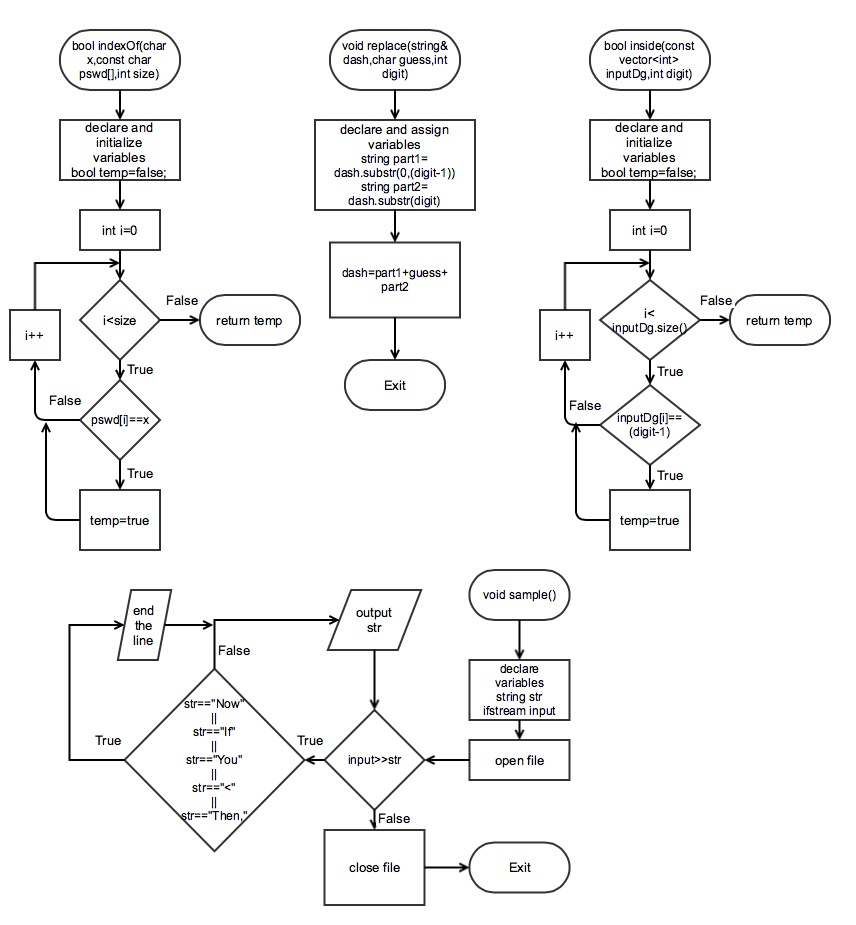
|  |  |  |  |
| --- | --- | --- | --- |
| vector | vector<char> | vector<char> nExist | 322 |
| Sorting | Sorting | for(int i=0;i<5;i++) {  for(int j=i+1;j<6;j++) {  if(nExist[i]<nExist[j]) {  char temp=nExist[i];  nExist[i]=nExist[j];  nExist[j]=temp;  }  }  } | 332 |
| Random number | char | srand(static\_cast<unsigned short>(time(0)));  int temp=rand()%numPyr; | 65 |

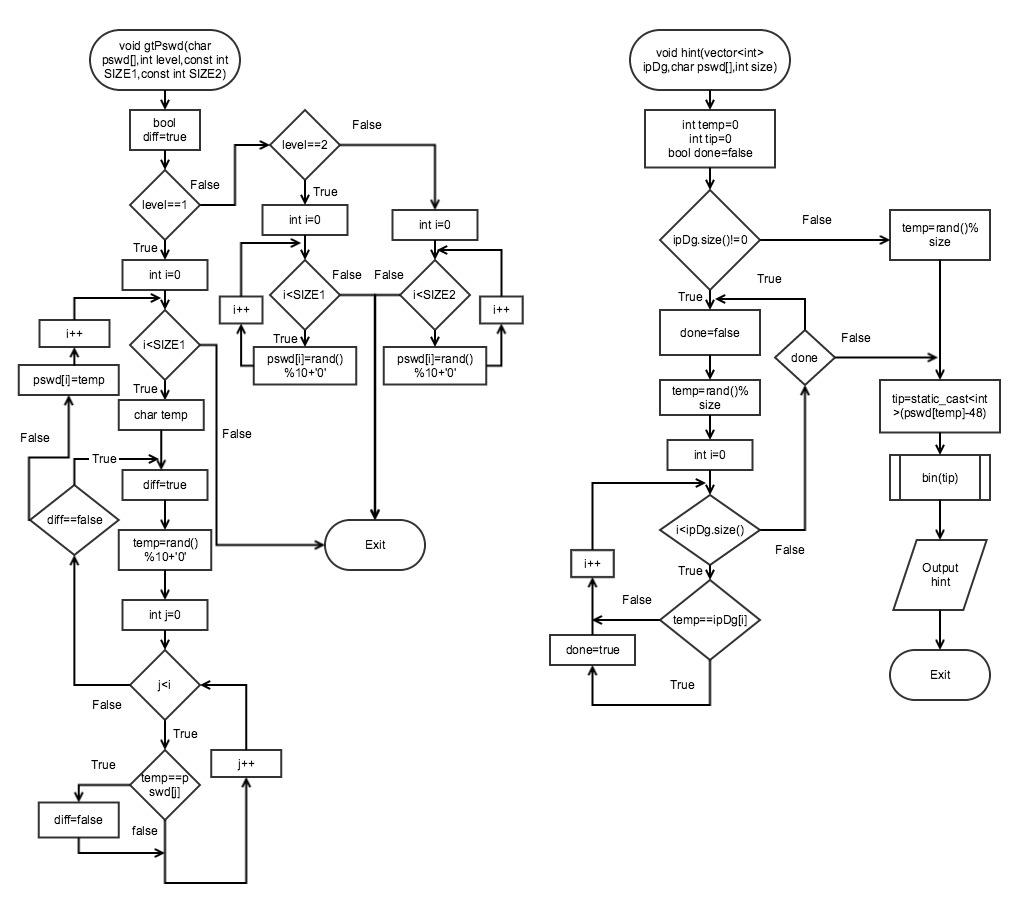
1. **Flowchart (1)main flowchart(3 pages)**

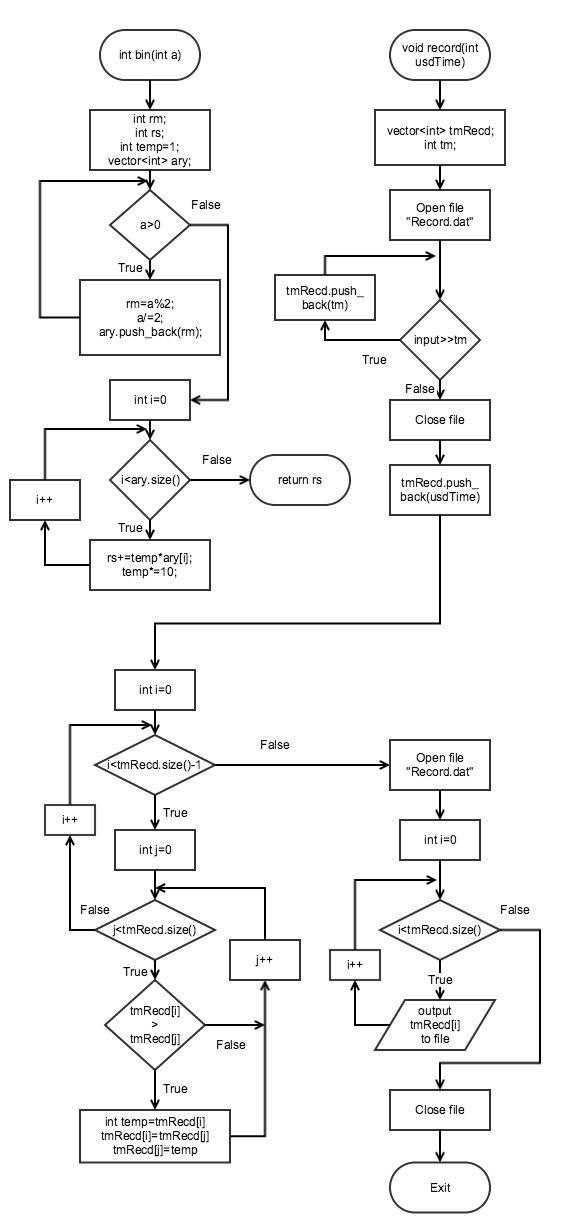


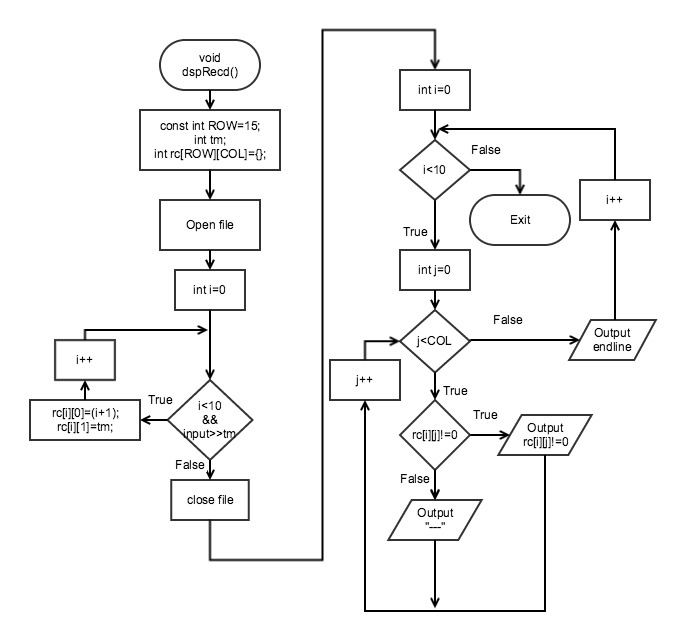
**(2) Function flowchart(5 pages)**







****

****

**12. Code**