Project 1: *Sì-Wŭ-Liù*

(a.k.a. Cee-lo)

Danziel Nguyen

CSC5-40772

Introduction:

Title: Sì-Wŭ-Liù

Sì-Wŭ-Liù is a basic dice gambling game played with three dice. It is often played in urban areas and was especially popular with Chinese-Americans. The American variant of the game "Cee-lo" has been referenced throughout the years in rap and hip-hop, aiding in the popularity of the game.

Sì-Wŭ-Liù can be played either with a banker or not. The game is generally most often played with a banker, and so the variant shown in my program is done with a predetermined banker with a single player. Similarly to craps, it uses a point system for dice rolls along with specific roll combinations. The banker normally changes from player to player but in this program it will not. The banker sets up an initial stake or "bank" and the other players will bet until the entire bank is covered or the players do not wish to bet further. If any portion of the bank is not covered, the uncovered amount is given back to the banker and the game proceeds with the stakes that were covered.

The banker rolls the dice once bets have been made. When the dice are rolled, four types of outcomes are possible:

If the banker rolls a 4, 5, and 6, called 4-5-6 straight kill, or if all three dice are the same (triples), or if the banker rolls a non-6 pair with an individual 6, it is an automatic win and the banker collects all bets.

If the banker rolls a 1, 2, and 3, called 1-2-3 straight loss, or if the banker rolls a non-1 pair with an individual 1, it is an automatic loss and the players break the bank.

If the banker rolls a pair and a single, the single dice is the banker's "point".

If the banker does not roll any of the above combinations, they roll again until they do so.

If the banker rolls a point, the players roll the dice to decide winners individually against the banker's point.

The outcomes are the same except players do not automatically win with a non-6 pair with a single 6 and they do not automatically lose with a non-1 pair with a single 1.

If the player rolls a point, the point is compared to the banker's point. The player wins with a higher point, lose with a lower point, if the points are tied the player rolls again.

The payouts differ based on variant however generally a 4-5-6 roll pays two times the bet, a roll of all 1s pay five times the bet, and regular non-1 triples pay three times the bet.

Program Summary:

Lines: about 450

Variables: 19

Libraries: 9

User Functions: 7

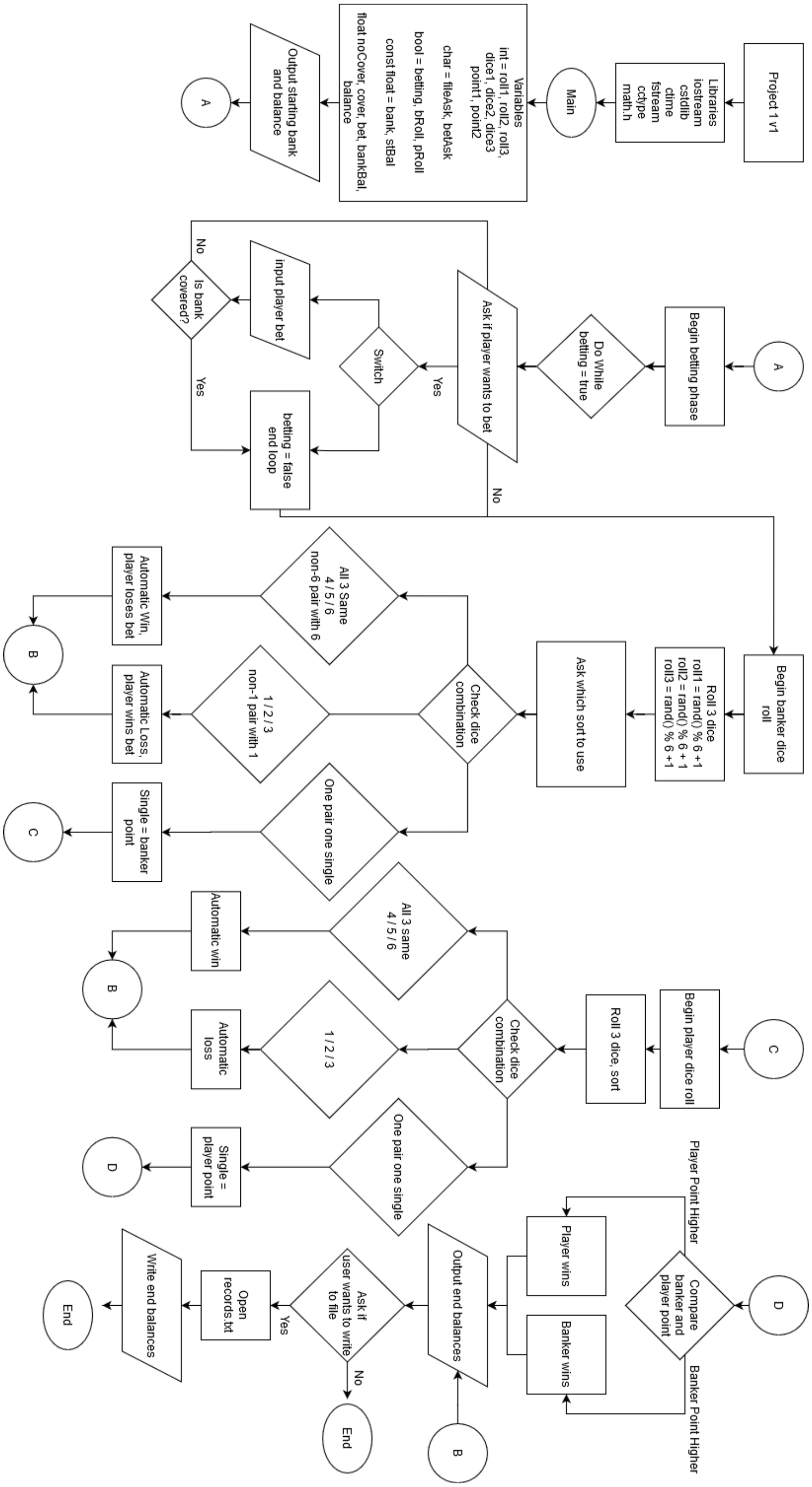
Pseudocode:

```
Libraries:
        iostream
        cstdlib
        ctime
        fstream
        cctype
        math.h
        algorithm
        vector
        string
Variables:
        Static const int: size
        Int: bDice[], point1, point2, pickSort
        char: betAsk
        bool: betting, bRoll, pRoll, logResult
        const float: bank, stBal
        float: noCover, cover, bet, bankbal, balance
        vector: pDice
        string: pName
Display starting bank and player balance
```

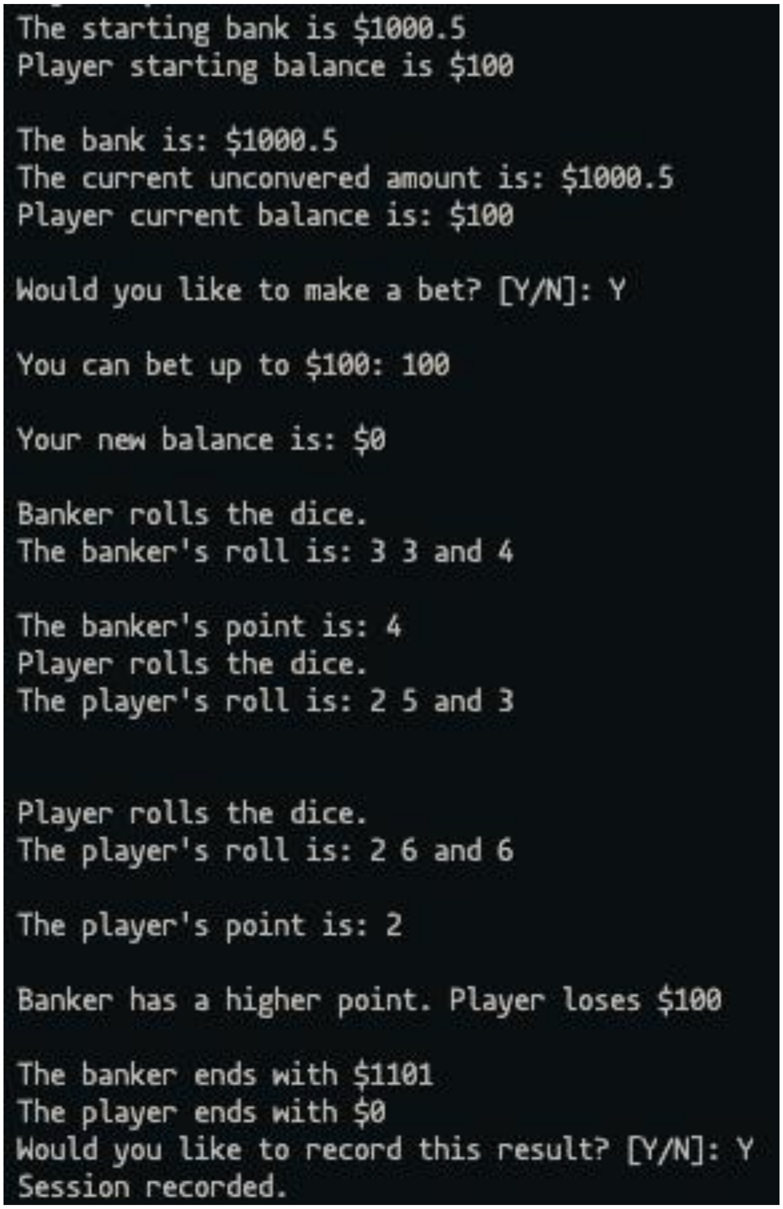
```
Roll banker's dice
Begin betting phase
Ask if the player wants to bet
If no: end
If yes: ask how much
If bank is covered, end betting phase
If bank is not covered, return to start of betting phase
    Sort dice from least to greatest
Check dice combinations:
    Three of the same , 4 5 6, non-6 pair with 6 are automatic
    win 1 2 3, non-1 pair with 1 are automatic lose
    Pair with single => single is the point
    If none of these, roll again
    If there is a point, set that point for the banker (point1)
Roll player's dice
Check dice combinations:
    1 1 1 automatic win, bet * 5
    Three of the same automatic win, bet * 3
    4 5 6 automatic win, bet *2
    1 2 3 automatic lose
    Pair with single => single is the point
    If none of these, roll again
    If there is a point, set that point for the player (point2)
If there are two points, compare them
    If the banker is higher, banker wins
    If the player is higher, player wins
```

```
Display the final bank balance and player balance
 Ask if the player wants to output the result to file
    If no, end program
    If yes, open records.txt
        Write end balances to file
        Close records.txt
End program
```

Flowchart:


Project 1 v1
Libraries
iostream
cstdlib
ctime
fstream
cctype
math.h
Main
Variables
int = roll1, roll2, roll3, dice1, dice2, dice3
point1, point2
char = fileAsk, betAsk
bool = betting, bRoll, pRoll
const float = bank, stBal
float noCover, cover, bet, bankBal, balance
Output starting bank and balance
A
A
Begin betting phase
Do While betting = true
Ask if player wants to bet
Yes
No
Switch
input player bet
Is bank covered?
No
Yes
betting = false
end loop
Begin banker dice roll
Roll 3 dice
roll1 = rand() % 6 + 1
roll2 = rand() % 6 + 1
roll3 = rand() % 6 + 1
Ask which sort to use
Check dice combination
All 3 Same
4 / 5 / 6
non-6 pair with 6
Automatic Win, player loses bet
1 / 2 / 3
non-1 pair with 1
Automatic Loss, player wins bet
B
One pair one single
Single = banker point
C
C
Begin player dice roll
Roll 3 dice, sort
Check dice combination
All 3 same
4 / 5 / 6
Automatic win
1 / 2 / 3
Automatic loss
B
One pair one single
Single = player point
D
D
Compare banker and player point
Player Point Higher
Player wins
Banker Point Higher
Banker wins
Output end balances
B
Ask if user wants to write to file
Yes
No
End
Open records.txt
Write end balances
End

Input and Output Example



```
The starting bank is $1000.5
Player starting balance is $100

The bank is: $1000.5
The current unconvered amount is: $1000.5
Player current balance is: $100

Would you like to make a bet? [Y/N]: Y

You can bet up to $100: 100

Your new balance is: $0

Banker rolls the dice.
The banker's roll is: 3 3 and 4

The banker's point is: 4
Player rolls the dice.
The player's roll is: 2 5 and 3


Player rolls the dice.
The player's roll is: 2 6 and 6

The player's point is: 2

Banker has a higher point. Player loses $100

The banker ends with $1101
The player ends with $0
Would you like to record this result? [Y/N]: Y
Session recorded.
```

Cross Reference from Project 1

You are to fill-in with where located in code

Chapter	Section	Topic	Where Line #"s	Pts	Notes
2	2	cout	59		
	3	libraries	15	5	iostream, iomanip, cmath, cstdlib, fstream, string, ctime
	4	variables/literals	44		No variables in global area, failed project!
	5	Identifiers	44		
	6	Integers	45	1	
	7	Characters	46	1	
	8	Strings	52	1	
	9	Floats No Doubles	50	1	Using doubles will fail the project, floats OK!
	10	Bools	47	1	
	11	Sizeof *****			
	12	Variables 7 characters or less	44		All variables <= 7 characters
	13	Scope ***** No Global Variables			
	14	Arithmetic operators	105		
	15	Comments 20%+		2	Model as pseudo code
	16	Named Constants	49		All Local, only Conversions/Physics/Math in Global area
	17	Programming Style ***** Emulate			Emulate style in book/in class repositiory
3	1	cin	56		
	2	Math Expression	105		
	3	Mixing data types ****			
	4	Overflow/Underflow ****			
	5	Type Casting		1	
	6	Multiple assignment *****			
	7	Formatting output	62	1	
	8	Strings	59	1	
	9	Math Library	21	1	All libraries included have to be used
	10	Hand tracing ******			
4	1	Relational Operators	85		
	2	if	146	1	Independent if
	4	If-else	95	1	
	5	Nesting	208	1	
	6	If-else-if	208	1	

	7	Flags *****			
	8	Logical operators	95	1	
	11	Validating user input	85	1	
	13	Conditional Operator	457	1	
	14	Switch	93	1	
5	1	Increment/Decrement	390	1	
	2	While	85	1	
	5	Do-while	74	1	
	6	For loop	390	1	
	11	Files input/output both		2	
	12	No breaks in loops ******			Failed Project if included
****** Not	required to	show	Total	30	

Cross Reference for Project 2

You are to fill-in with where located in code

Chapter	Section	Topic	Where Line #"s	Pts	Notes
6		Functions	389		
	3	Function Prototypes	32	4	Always use prototypes
	5	Pass by Value		4	
	8	return	459	4	A value from a function
	9	returning boolean	459	4	
	10	Global Variables		XXX	Do not use global variables -100 pts
	11	static variables	45	4	
	12	defaulted arguments	426	4	
	13	pass by reference	389	4	
	14	overloading	396	5	

	15	exit() function	137	4	
7		Arrays	46		
	1 to 6	Single Dimensioned Arrays	46	3	
	7	Parallel Arrays		2	
	8	Single Dimensioned as Function A	rguments 389	2	
	9	2 Dimensioned Arrays		2	Emulate style in book/in class repositiory
	12	STL Vectors	52	2	
		Passing Arrays to and from Functio	ns 389	5	
		Passing Vectors to and from Funct	ons 396	5	
8		Searching and Sorting Arrays	415		
	3	Bubble Sort	415	4	
	3	Selection Sort	426	4	
	1	Linear or Binary Search		4	
****** Not	requir ed to	show	Total	70	Other 30 points from Proj 1 first sheet tab

i