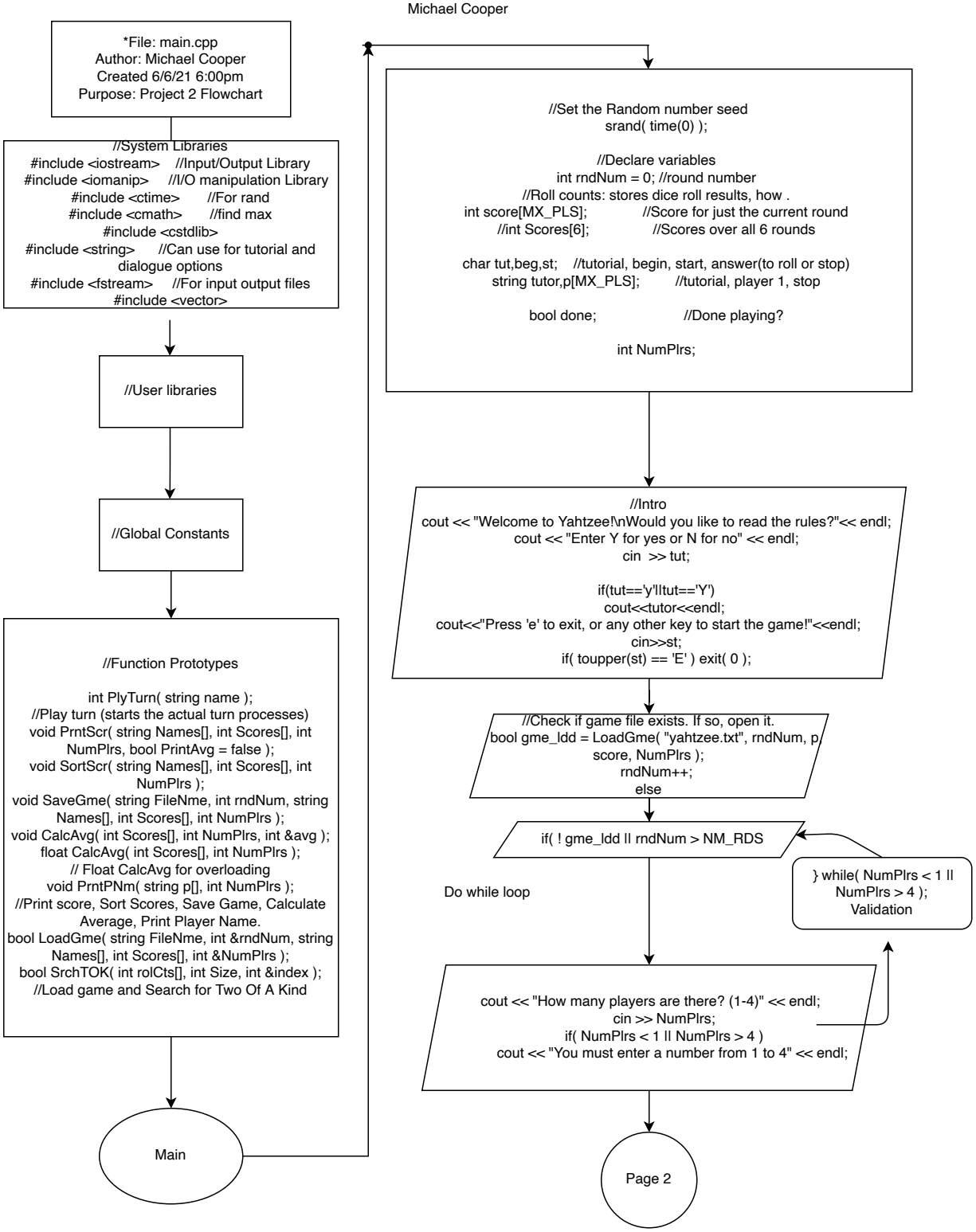
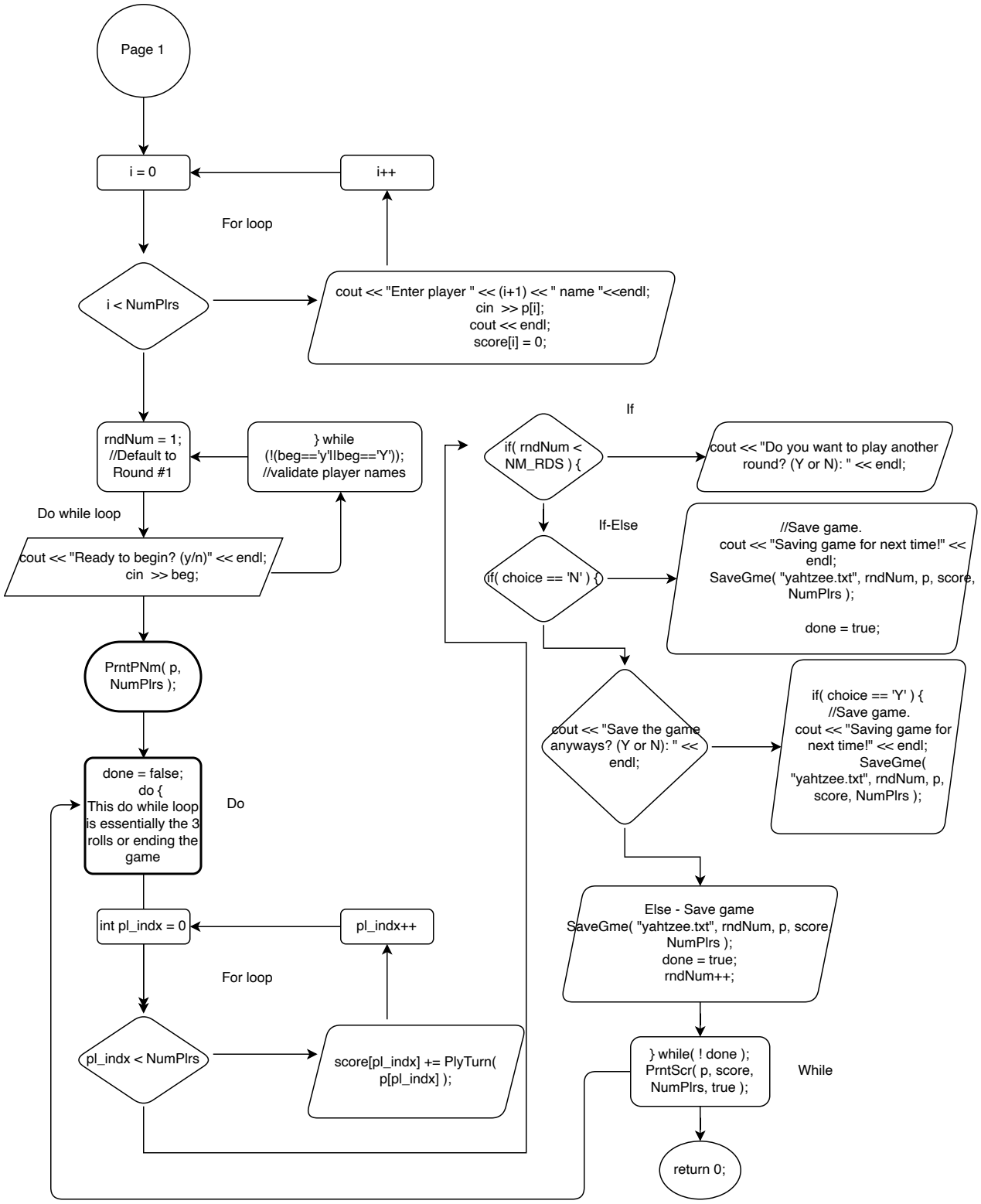
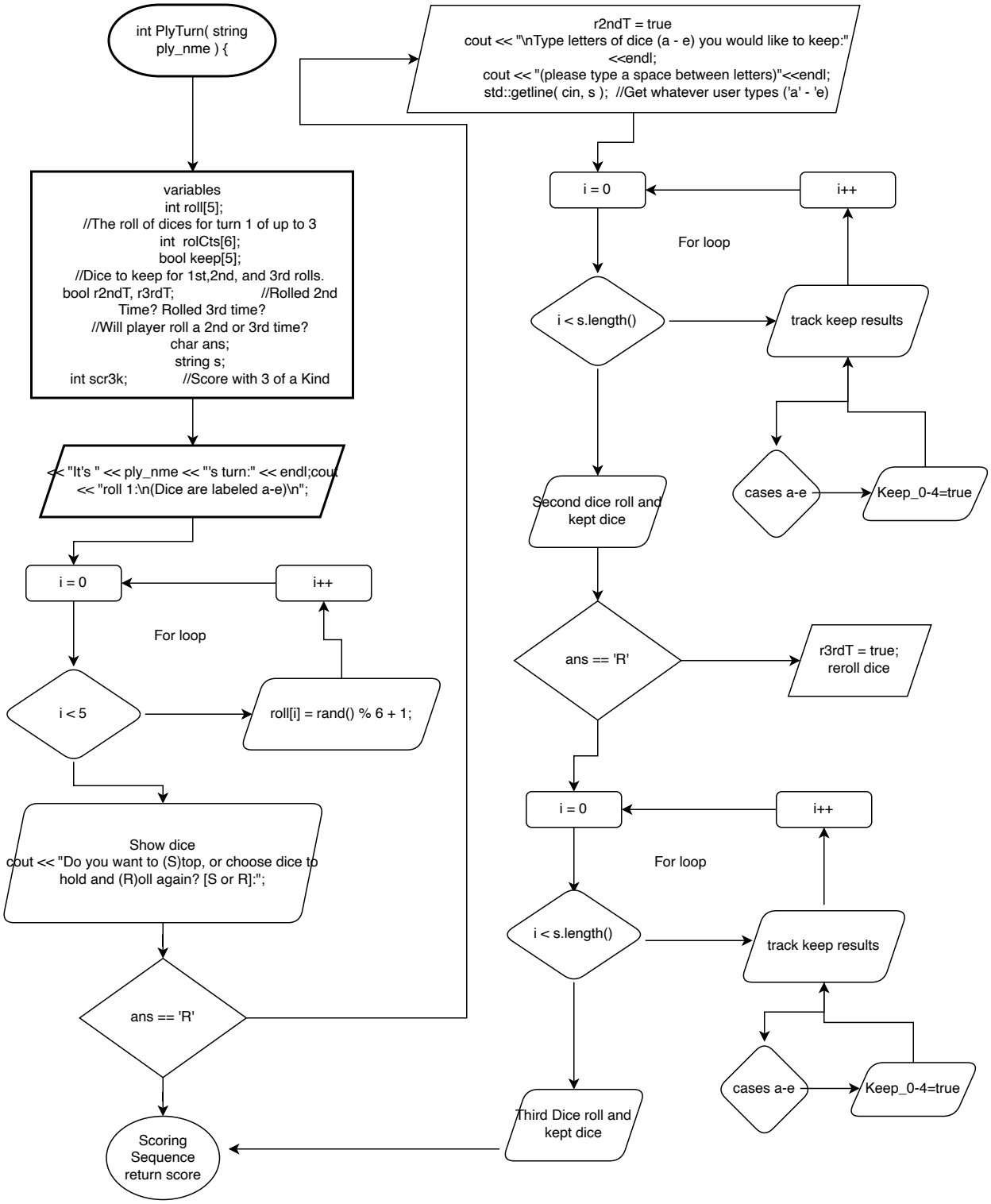
Project 2 FlowChart

Michael Cooper

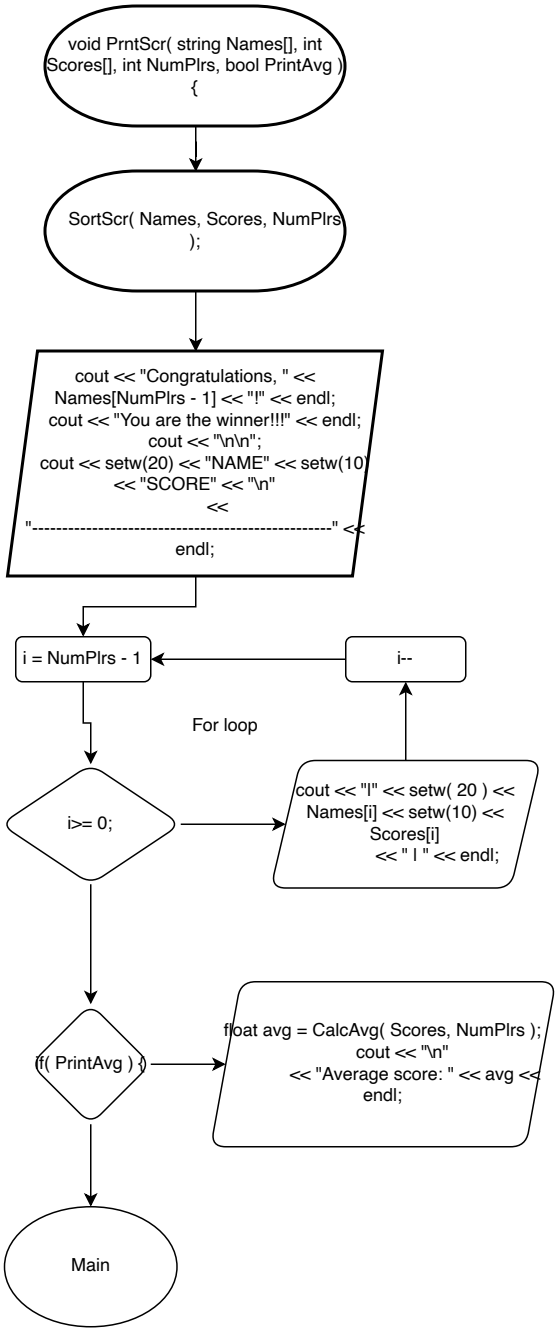


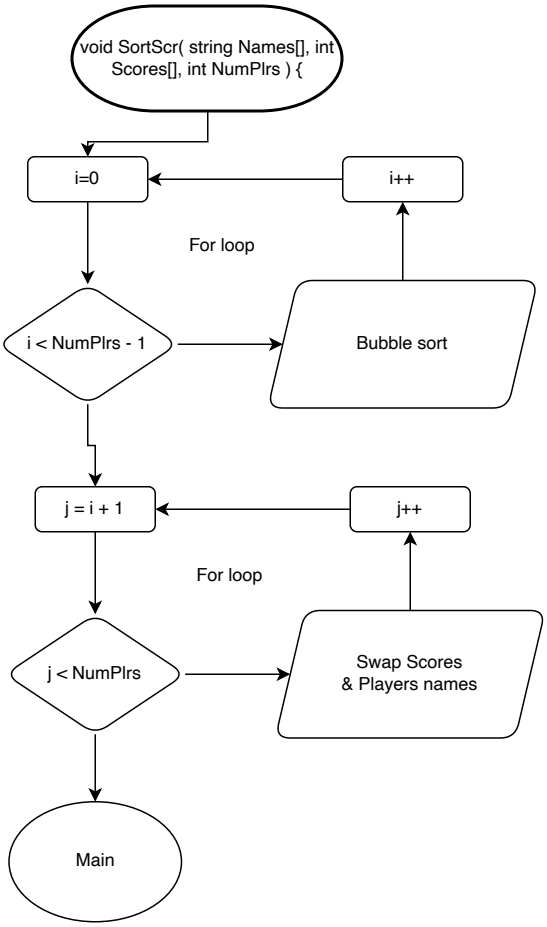

Page 1
i = 0
i++
For loop
i < NumPlrs
cout << "Enter player " << (i+1) << " name "<<endl;
cin >> p[i];
cout << endl;
score[i] = 0;
rndNum = 1;
//Default to
Round #1
} while
(!(beg=='y'||beg=='Y'));
//validate player names
Do while loop
cout << "Ready to begin? (y/n)" << endl;
cin >> beg;
PrntPNm(p,
NumPlrs);
done = false;
do {
This do while loop
is essentially the 3
rolls or ending the
game
Do
int pl_indx = 0
pl_indx++
For loop
pl_indx < NumPlrs
score[pl_indx] += PlyTurn(
p[pl_indx]);
If
if(rndNum <
NM_RDS) {
cout << "Do you want to play another
round? (Y or N): " << endl;
If-Else
if(choice == 'N') {
//Save game.
cout << "Saving game for next time!" <<
endl;
SaveGme("yahtzee.txt", rndNum, p, score,
NumPlrs);
done = true;
cout << "Save the game
anyways? (Y or N): " <<
endl;
if(choice == 'Y') {
//Save game.
cout << "Saving game for
next time!" << endl;
SaveGme(
"yahtzee.txt", rndNum, p,
score, NumPlrs);
Else - Save game
SaveGme("yahtzee.txt", rndNum, p, score,
NumPlrs);
done = true;
rndNum++;
} while(! done);
PrntScr(p, score,
NumPlrs, true);
While
return 0;

Functions

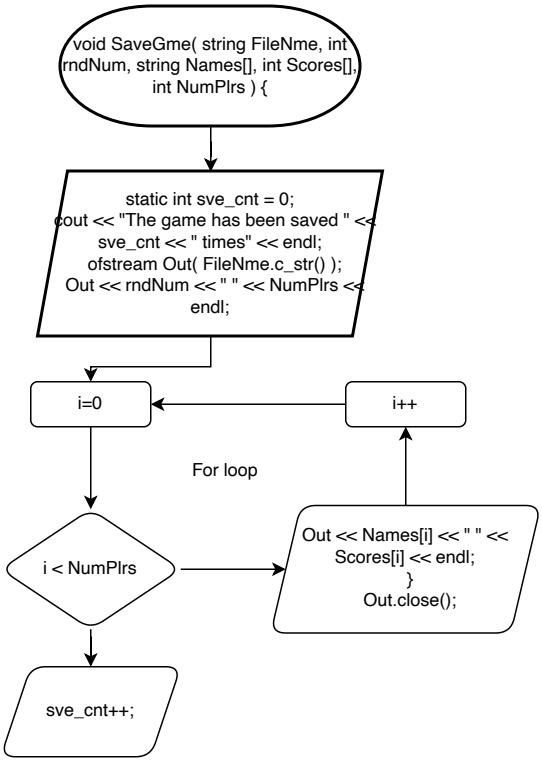

int PlyTurn(string ply_nme) {
variables
int roll[5];
//The roll of dices for turn 1 of up to 3
int rolCts[6];
bool keep[5];
//Dice to keep for 1st,2nd, and 3rd rolls.
bool r2ndT, r3rdT; //Rolled 2nd Time? Rolled 3rd time?
//Will player roll a 2nd or 3rd time?
char ans;
string s;
int scr3k; //Score with 3 of a Kind
<< "It's " << ply_nme << "'s turn:" << endl;cout << "roll 1:\n(Dice are labeled a-e)\n";
i = 0
i++
For loop
i < 5
roll[i] = rand() % 6 + 1;
Show dice
cout << "Do you want to (S)top, or choose dice to hold and (R)oll again? [S or R]:";
ans == 'R'
Scoring Sequence return score
r2ndT = true
cout << "\nType letters of dice (a - e) you would like to keep:" <<endl;
cout << "(please type a space between letters)"<<endl;
std::getline(cin, s); //Get whatever user types ('a' - 'e)
i = 0
i++
For loop
i < s.length()
track keep results
cases a-e
Keep_0-4=true
Second dice roll and kept dice
ans == 'R'
r3rdT = true; reroll dice
i = 0
i++
For loop
i < s.length()
track keep results
cases a-e
Keep_0-4=true
Third Dice roll and kept dice

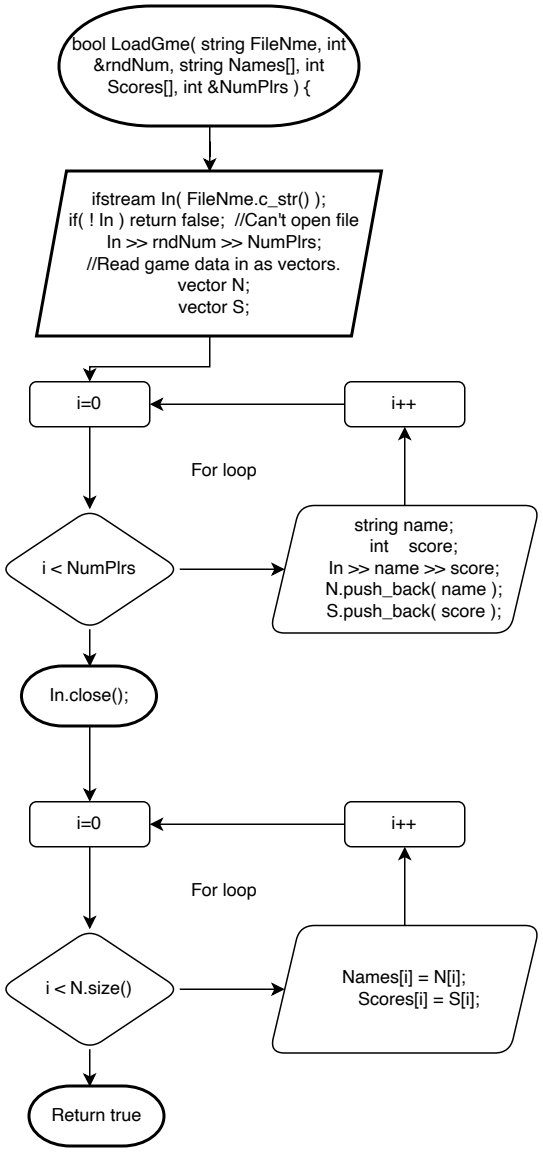
Functions


void PrntScr(string Names[], int Scores[], int NumPlrs, bool PrintAvg) {
SortScr(Names, Scores, NumPlrs);
cout << "Congratulations, " << Names[NumPlrs - 1] << "!" << endl;
cout << "You are the winner!!!" << endl;
cout << "\n\n";
cout << setw(20) << "NAME" << setw(10) << "SCORE" << "\n"
<<
"----------" << endl;
i = NumPlrs - 1
i--
For loop
i>= 0;
cout << "|" << setw(20) << Names[i] << setw(10) << Scores[i]
<< " | " << endl;
if(PrintAvg) {
float avg = CalcAvg(Scores, NumPlrs);
cout << "\n"
<< "Average score: " << avg << endl;
Main

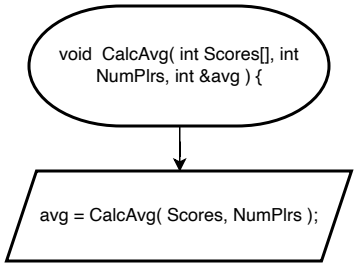

void SortScr(string Names[], int Scores[], int NumPlrs) {
i=0
i++
For loop
i < NumPlrs - 1
Bubble sort
j = i + 1
j++
For loop
j < NumPlrs
Swap Scores
& Players names
Main

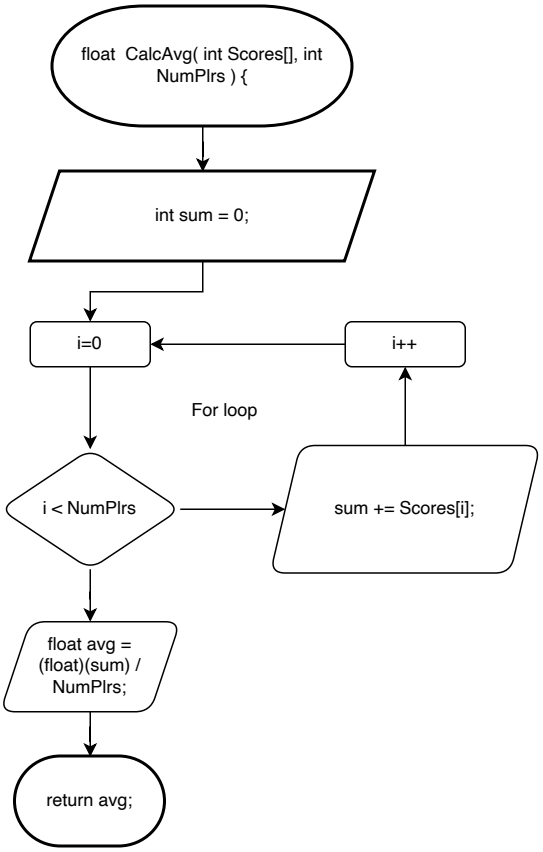
Functions

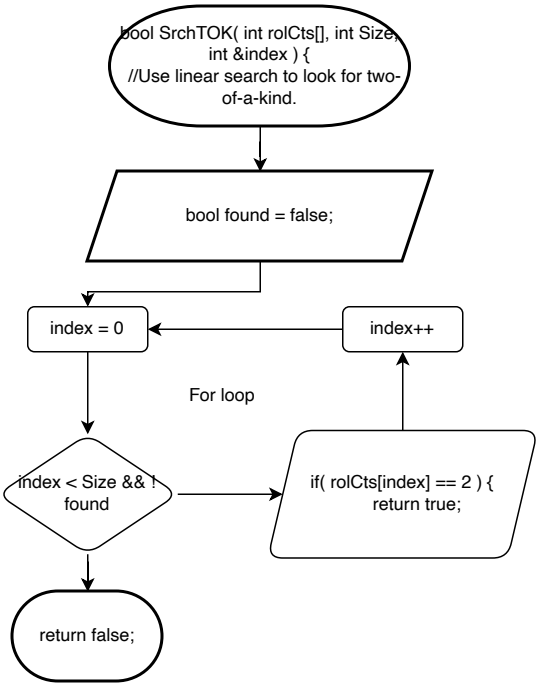

void SaveGme(string FileNme, int rndNum, string Names[], int Scores[], int NumPlrs) {
static int sve_cnt = 0;
cout << "The game has been saved " << sve_cnt << " times" << endl;
ofstream Out(FileNme.c_str());
Out << rndNum << " " << NumPlrs << endl;
i=0
i++
For loop
i < NumPlrs
Out << Names[i] << " " << Scores[i] << endl;
}
Out.close();
sve_cnt++;


bool LoadGme(string FileNme, int &rndNum, string Names[], int Scores[], int &NumPlrs) {
ifstream In(FileNme.c_str());
if(! In) return false; //Can't open file
In >> rndNum >> NumPlrs;
//Read game data in as vectors.
vector N;
vector S;
i=0
i++
For loop
i < NumPlrs
string name;
int score;
In >> name >> score;
N.push_back(name);
S.push_back(score);
In.close();
i=0
i++
For loop
i < N.size()
Names[i] = N[i];
Scores[i] = S[i];
Return true

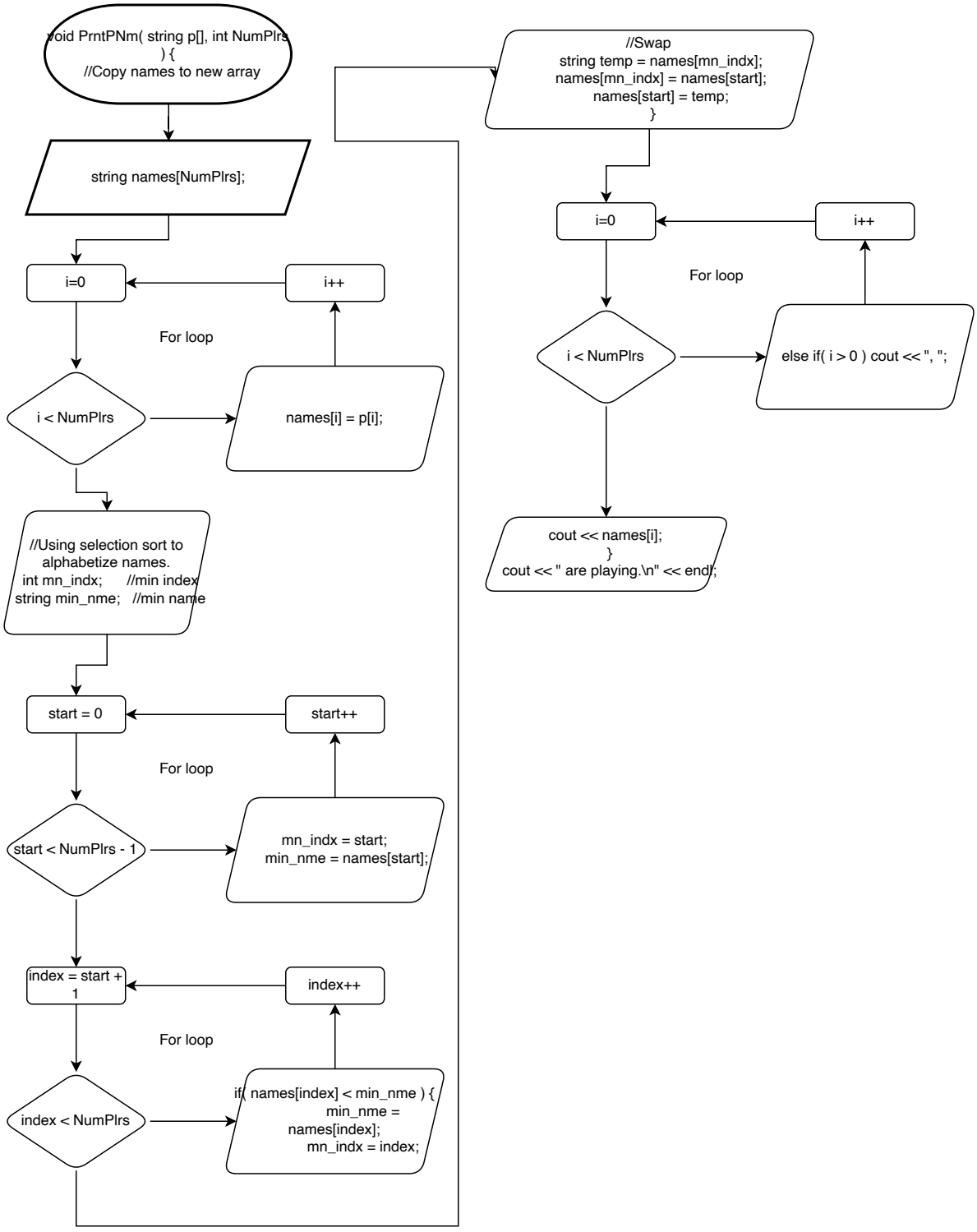
Functions


void CalcAvg(int Scores[], int NumPlrs, int &avg) {
avg = CalcAvg(Scores, NumPlrs);


float CalcAvg(int Scores[], int NumPlrs) {
int sum = 0;
i=0
i++
For loop
i < NumPlrs
sum += Scores[i];
float avg = (float)(sum) / NumPlrs;
return avg;


bool SrchTOK(int rolCts[], int Size, int &index) {
//Use linear search to look for two-of-a-kind.
bool found = false;
index = 0
index++
For loop
index < Size && !found
if(rolCts[index] == 2) {
return true;
return false;

Functions


void PrntPNm(string p[], int NumPlrs) {
//Copy names to new array
string names[NumPlrs];
i=0
i++
For loop
i < NumPlrs
names[i] = p[i];
//Using selection sort to alphabetize names.
int mn_indx; //min index
string min_nme; //min name
start = 0
start++
For loop
start < NumPlrs - 1
mn_indx = start;
min_nme = names[start];
index = start + 1
index++
For loop
index < NumPlrs
if(names[index] < min_nme) {
min_nme = names[index];
mn_indx = index;
//Swap
string temp = names[mn_indx];
names[mn_indx] = names[start];
names[start] = temp;
}
i=0
i++
For loop
i < NumPlrs
else if(i > 0) cout << ", ";
cout << names[i];
}
cout << " are playing.\n" << endl;