```cpp
// Assignment2ExtendedA1.cpp : This file contains the 'main' function. Program execution begins and ends there.
//

#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

struct player {
    string fullName;
    string fName;
    string lName;
    char marker;
    int stats[3] = { 0, 0, 0 };
};

void outputHeader();
int stringToInt(string);
int intIsValid(string, int, int);
player* getPlayers(int*);
void getNames(int*, player*);
void getBoardDimensions(int*);
char** initializeBoard(int*);
void printBoard(int*, char**);
char isWin(int*, char**, player*, char);
void itsaTie(int*, player*, char**);
string choiceIsValid(string, int*);
void canMove(string, int*, char**, char);
void makeMove(char**, player*, int*, char);
char switchTurn(char, int*, player*);
void outputStats(player*, int*);


int main()
{
    outputHeader();

    int settings[4];
    char** board;
    player* playerInfo = getPlayers(settings);
    char playAgain = 'y';
    char turn = 'a';
    char whoStarted;
    char winner;
    bool isTie = false;
    int count;

    getNames(settings, playerInfo);


```

```
52      while (playAgain == 'y' || playAgain == 'Y')
53      {
54          getBoardDimensions(settings);
55          board = initializeBoard(settings);
56
57          int row = settings[0];
58          int col = settings[1];
59          count = 1;
60          winner = ' ';
61          whoStarted = turn;
62
63          cout << "Start game!!!" << endl << endl;
64
65          while (winner == ' ' && !isTie)
66          {
67              printBoard(settings, board);
68              makeMove(board, playerInfo, settings, turn);
69              winner = isWin(settings, board, playerInfo, turn);
70
71              if (winner != ' ')
72              {
73                  printBoard(settings, board);
74              }
75
76              if (count >= row * col)
77              {
78                  itsaTie(settings, playerInfo, board);
79                  isTie = true;
80              }
81
82              turn = switchTurn(turn, settings, playerInfo);
83
84              count++;
85          }
86
87          if (isTie)
88          {
89              whoStarted = switchTurn(whoStarted, settings, playerInfo);
90              turn = whoStarted;
91              isTie = false;
92          }
93          else
94          {
95              turn = switchTurn(winner, settings, playerInfo);
96          }
97
98          outputStats(playerInfo, settings);
99
100         cout << "Continue? (y/n) >> ";
101         cin >> playAgain;
102         cout << endl;
103     }
```

```
104
105     cout << "Thanks for playing!" << endl;
106 }
107
108 /************************************************************
109 *   THIS FUNCTION OUTPUTS A HEADER FOR THE GAME             *
110 *                                                           *
111 ************************************************************/
112
113 void outputHeader()
114 {
115     cout << "/************************************************************\ " << endl;
116     cout << "*                                                            * " << endl;
117     cout << "*                    TIC TAC TOE II                          * " << endl;
118     cout << "*                                                            * " << endl;
119     cout << "\************************************************************/ " << endl << endl;
120
121     cout << "WELCOME USERS!" << endl << endl;
122     cout << "# TO GET STARTED ENTER THE NUMBER OF PLAYERS." << endl;
123     cout << "# ENTER YOUR FIRST AND LAST NAMES." << endl;
124     cout << "# ONCE YOU FIGURE OUT THE BOARD DIMENSIONS THE GAME WILL START." << endl;
125     cout << "# ENTER A LETTER AND A NUMBER TO MAKE YOUR MOVE. " << endl << endl;
126     cout << "ENJOY THE GAME!!!" << endl << endl;
127 }
128
129 /************************************************************
130 *      THIS FUNCTION CONVERTS A STRING TO AN INTEGER         *
131 *                                                           *
132 ************************************************************/
133
134 int stringToInt(string str)
135 {
136     int num = 0;
137     int decimal;
138     int exponent = 0;
139
140     for (int i = str.length() - 1; i >= 0; i--)
141     {
142         decimal = 1;
143
144         for (int i = 0; i < exponent; i++)
145         {
146             decimal *= 10;
147         }
148
149         int digit = str[i] - '0';
```

```
150         num += digit * decimal;
151         exponent++;
152     }
153
154     return num;
155 }
156
157 /*************************************************************
158 *    THIS FUNCTION CHECKS TO SEE IF AN INT USER INPUT IS     *
159 *                          VALID                            *
160 *************************************************************/
161
162 int intIsValid(string check, int min, int max)
163 {
164     bool valid = false;
165     string tryAgain;
166
167     int checking = stringToInt(check);
168
169     while (!valid)
170     {
171         if (checking < min || checking > max)
172         {
173             cout << "You Should Try a Number Between " << min << " & " << max <<  ↵
                  " >> ";
174             cin >> tryAgain;
175             cout << endl;
176
177             check = tryAgain;
178             checking = stringToInt(check);
179         }
180         else
181         {
182             valid = true;
183         }
184     }
185
186     cout << endl;
187
188     return checking;
189 }
190
191 /*************************************************************
192 *   THIS FUNCTION GETS THE NUMBER OF PLAYERS FOR THE GAME    *
193 *                                                            *
194 *************************************************************/
195
196 player* getPlayers(int* gameSettings)
197 {
198     string input;
199
200     cout << "Please enter the number of players for this game >> ";
```

```
201     cin >> input;
202     cout << endl;
203
204     gameSettings[2] = intIsValid(input, 2, 5);
205     gameSettings[3] = 0;
206
207     player* playerInfo = new player[gameSettings [2]];
208
209     char peice = 'a';
210
211     for (int i = 0; i < gameSettings[2]; i++)
212     {
213         playerInfo[i].marker = peice;
214         peice++;
215     }
216
217     return playerInfo;
218 }
219
220 /************************************************************
221 *      THIS FUNCTION GETS THE NAMES OF ALL THE PLAYERS       *
222 *                                                            *
223 ************************************************************/
224
225 void getNames(int* gameSettings, player* playerInfo)
226 {
227     for (int i = 0; i < gameSettings[2]; i++)
228     {
229         cout << "Player " << i + 1 << ", please enter your first and last name >> ↵
            ";
230         cin >> playerInfo[i].fName >> playerInfo[i].lName;
231         cout << endl;
232
233         playerInfo[i].fName[0] = toupper(playerInfo[i].fName[0]);
234         playerInfo[i].lName[0] = toupper(playerInfo[i].lName[0]);
235         playerInfo[i].fullName = playerInfo[i].fName + " " + playerInfo[i].lName;
236     }
237
238     cout << endl;
239 }
240
241 /************************************************************
242 *     THIS FUNCTION GETS NUMBER OF ROWS AND COLUMNS FOR      *
243 *                          THE BOARD                         *
244 ************************************************************/
245
246 void getBoardDimensions(int* gameSettings)
247 {
248     string input;
249
250     cout << "Please enter the number of rows >> ";
251     cin >> input;
```

```
252     cout << endl;
253
254     gameSettings[0] = intIsValid(input, 3, 11);
255
256     cout << "Please enter the number of columns >> ";
257     cin >> input;
258     cout << endl;
259
260     gameSettings[1] = intIsValid(input, 3, 16);
261 }
262
263 /*************************************************************
264 *        THIS FUNCTION EMPTIES ALL THE BOARD SPACES          *
265 *                                                            *
266 *************************************************************/
267
268 char** initializeBoard(int* gameSettings)
269 {
270     int row = gameSettings[0];
271     int col = gameSettings[1];
272
273     char** board = new char*[row];
274
275     for (int i = 0; i < row; i++)
276     {
277         board[i] = new char[col];
278     }
279
280     for (int i = 0; i < row; i++)
281     {
282         for (int j = 0; j < col; j++)
283         {
284             board[i][j] = ' ';
285         }
286     }
287
288     return board;
289 }
290
291 /*************************************************************
292 *          THIS FUNCTION PRINTS OUT THE GAME BOARD           *
293 *                                                            *
294 *************************************************************/
295
296 void printBoard(int* gameSettings, char** board)
297 {
298     int isRow = 1;
299     int size = gameSettings[0] * gameSettings[1];
300     int row = gameSettings[0] * 2 + 3;
301     int col = gameSettings[1];
302     int rowElement = 0;
303     char letter = 'A';
```

```
304
305     for (int i = 0; i < row; i++)
306     {
307         for (int j = 0; j < col; j++)
308         {
309             if (i == 0 || i == row - 1)
310             {
311                 if (j == 0)
312                 {
313                     cout << "    " << j + 1;
314                 }
315                 else if (j >= 9)
316                 {
317                     cout << "  " << j + 1;
318                 }
319                 else
320                 {
321                     cout << "   " << j + 1;
322                 }
323             }
324             else
325             {
326                 if (isRow % 2 == 0)
327                 {
328                     if (j == 0)
329                     {
330                         cout << letter << " ";
331                     }
332
333                     cout << ": " << board[rowElement][j] << " ";
334                 }
335                 else
336                 {
337                     if (j == 0)
338                     {
339                         cout << "  ";
340                     }
341
342                     cout << " ---";
343                 }
344
345             }
346
347         }
348
349         if (isRow % 2 == 0 && i != row - 1)
350         {
351             cout << ": " << letter;
352
353             letter++;
354             rowElement++;
355         }
```

```
356
357         if (i != 0)
358         {
359             isRow++;
360         }
361
362         cout << endl;
363     }
364
365     cout << endl;
366 }
367
368 /************************************************************
369 *        THIS FUNCTION CHECKS THE BOARD FOR A WIN            *
370 *                                                            *
371 ************************************************************/
372
373 char isWin(int* gameSettings, char** board, player* playerInfo, char turn)
374 {
375     for (int i = 0; i < gameSettings[0]; i++)
376     {
377         for (int j = 0; j < gameSettings[1] - 2; j++)
378         {
379             if (board[i][j] == board[i][j + 1] && board[i][j + 1] == board[i][j + ↵
                2] && turn == board[i][j])
380             {
381                 board[i][j] = board[i][j + 1] = board[i][j + 2] = toupper(turn);
382
383                 for (int k = 0; k < *(gameSettings + 2); k++)
384                 {
385                     if (playerInfo[k].marker == turn)
386                     {
387                         cout << playerInfo[k].fName << " wins!!!" << endl <<      ↵
                         endl;
388
389                         playerInfo[k].stats[0] += 1;
390                     }
391                     else
392                     {
393                         playerInfo[k].stats[1] += 1;
394                     }
395                 }
396
397                 gameSettings[3] += 1;
398
399                 return turn;
400             }
401         }
402     }
403
404     for (int i = 0; i < *(gameSettings) - 2; i++)
405     {
```

```
406         for (int j = 0; j < *(gameSettings + 1); j++)
407         {
408             if (board[i][j] == board[i + 1][j] && board[i + 1][j] == board[i + 2]
                [j] && turn == board[i][j])
409             {
410                 board[i][j] = board[i + 1][j] = board[i + 2][j] = toupper(turn);
411
412                 for (int k = 0; k < *(gameSettings + 2); k++)
413                 {
414                     if (playerInfo[k].marker == turn)
415                     {
416                         cout << playerInfo[k].fName << " wins!!!" << endl <<
                        endl;
417
418                         playerInfo[k].stats[0] += 1;
419                     }
420                     else
421                     {
422                         playerInfo[k].stats[1] += 1;
423                     }
424                 }
425
426                 gameSettings[3] += 1;
427
428                 return turn;
429             }
430         }
431     }
432
433     for (int i = 0; i < *(gameSettings) - 2; i++)
434     {
435         for (int j = 0; j < *(gameSettings + 1) - 2; j++)
436         {
437             if (board[i][j] == board[i + 1][j + 1] && board[i + 1][j + 1] ==
                board[i + 2][j + 2] && turn == board[i][j])
438             {
439                 board[i][j] = board[i + 1][j + 1] = board[i + 2][j + 2] = toupper
                  (turn);
440
441                 for (int k = 0; k < *(gameSettings + 2); k++)
442                 {
443                     if (playerInfo[k].marker == turn)
444                     {
445                         cout << playerInfo[k].fName << " wins!!!" << endl <<
                        endl;
446
447                         playerInfo[k].stats[0] += 1;
448                     }
449                     else
450                     {
451                         playerInfo[k].stats[1] += 1;
452                     }
```

```
453                 }
454
455                 gameSettings[3] += 1;
456
457                 return turn;
458             }
459         }
460     }
461
462     for (int i = 0; i < *(gameSettings) - 2; i++)
463     {
464         for (int j = 2; j < *(gameSettings + 1); j++)
465         {
466             if (board[i][j] == board[i + 1][j - 1] && board[i + 1][j - 1] ==
                  board[i + 2][j - 2] && turn == board[i][j])
467             {
468                 board[i][j] = board[i + 1][j - 1] = board[i + 2][j - 2] = toupper
                      (turn);
469
470                 for (int k = 0; k < *(gameSettings + 2); k++)
471                 {
472                     if (playerInfo[k].marker == turn)
473                     {
474                         cout << playerInfo[k].fName << " wins!!!" << endl <<
                          endl;
475
476                         playerInfo[k].stats[0] += 1;
477                     }
478                     else
479                     {
480                         playerInfo[k].stats[1] += 1;
481                     }
482                 }
483
484                 gameSettings[3] += 1;
485
486                 return turn;
487             }
488         }
489     }
490
491     return ' ';
492 }
493
494 /*************************************************************
495 *           THIS FUNCTION DECLARES THE GAME A TIE            *
496 *                                                            *
497 *************************************************************/
498
499 void itsaTie(int* gameSettings, player* playerInfo, char** board)
500 {
501     for (int i = 0; i < gameSettings[2]; i++)
```

```
502     {
503         playerInfo[i].stats[2] += 1;
504     }
505
506     gameSettings[3] += 1;
507
508     cout << "It's a tie." << endl << endl;
509
510     printBoard(gameSettings, board);
511 }
512
513 /*************************************************************
514 *   THIS FUNCTION CHECKS THE USER'S INPUT AND MAKE SURE ITS  *
515 *                      WITHIN THE BOUNDS                     *
516 *************************************************************/
517
518 string choiceIsValid(string choice, int* gameSettings)
519 {
520     char max = 'A';
521     bool valid = false;
522     int count = 0;
523     string num;
524     int col;
525
526     for (int i = 0; i < *(gameSettings) - 1; i++)
527     {
528         max++;
529     }
530
531     while (!valid)
532     {
533         choice[0] = toupper(choice[0]);
534         num = choice.substr(1);
535         col = stringToInt(num);
536
537         if (choice[0] > max || choice[0] < 'A' || col > *(gameSettings + 1) ||
              col < 1)
538         {
539             cout << "Invalid move, please try again >> ";
540             cin >> choice;
541             cout << endl;
542         }
543         else
544         {
545             valid = true;
546         }
547     }
548
549     cout << endl;
550
551     return choice;
552 }
```

```
553
554 /************************************************************
555 *   THIS FUNCTION CHECKS TO SEE IF THE SPACE A PLAYER CHOSE  *
556 *                        IS NOT TAKEN                        *
557 ************************************************************/
558
559 void canMove(string move, int* gameSettings, char** board, char turn)
560 {
561     char max = 'A';
562     int count = 0;
563     bool valid = false;
564     string num;
565     int row;
566     int col;
567
568     for (int i = 0; i < *(gameSettings) - 1; i++)
569     {
570         max++;
571     }
572
573     while (!valid)
574     {
575         num = move.substr(1);
576         col = stringToInt(num) - 1;
577         char charRow = 'A';
578
579         while (charRow != move[0])
580         {
581             count++;
582             charRow++;
583         }
584
585         row = count;
586
587         if (board[row][col] != ' ')
588         {
589             cout << "This space is taken, choose another one >> ";
590             cin >> move;
591             cout << endl;
592
593             move = choiceIsValid(move, gameSettings);
594         }
595         else
596         {
597             valid = true;
598         }
599
600         count = 0;
601     }
602
603     board[row][col] = turn;
604 }
```

```
605
606 /************************************************************
607 *   THIS FUNCTION ASKS THE USER TO MAKE THEIR MOVE AND       *
608 *             CHANGES THE BOARD ACCORDINGLY                  *
609 ************************************************************/
610
611 void makeMove(char** board, player* playerInfo, int* gameSettings, char turn)
612 {
613     string choice;
614     int count = 0;
615
616     for (char c = 'a'; c <= 'e'; c++)
617     {
618         if (turn == c)
619         {
620             cout << (playerInfo + count)->fName << "(" << turn << "), please make
                    your move >> ";
621             cin >> choice;
622             cout << endl;
623         }
624
625         count++;
626     }
627
628     choice = choiceIsValid(choice, gameSettings);
629     canMove(choice, gameSettings, board, turn);
630 }
631
632 /************************************************************
633 *          THIS FUNCTION SWITCHES WHOSE TURN IT IS          *
634 *                                                           *
635 ************************************************************/
636
637 char switchTurn(char turn, int* gameSettings, player* playerInfo)
638 {
639     if (turn == playerInfo[gameSettings[2] - 1].marker)
640     {
641         return 'a';
642     }
643     else
644     {
645         turn++;
646     }
647
648     return turn;
649 }
650
651 /************************************************************
652 *       THIS FUNCTION OUTPUTS THE STATS FOR EACH PLAYER     *
653 *                                                           *
654 ************************************************************/
655
```

```
656 void outputStats(player* playerInfo, int* gameSettings)
657 {
658     int maxSpace = playerInfo[0].fullName.length();
659     string tableRow = "  ------ ------ ------";
660     string winLoss = " : WINS : LOSS : DRAW :";
661 
662     for (int i = 0; i < *(gameSettings + 2); i++)
663     {
664         if (playerInfo[i].fullName.length() > maxSpace)
665         {
666             maxSpace = playerInfo[i].fullName.length();
667         }
668     }
669 
670     maxSpace += 1;
671 
672     int rowLength = maxSpace + tableRow.length();
673     int winLength = maxSpace + winLoss.length();
674 
675     cout << "Total game(s) played: " << gameSettings[3] << endl << endl;
676     cout << setw(rowLength) << tableRow << endl;
677     cout << setw(winLength) << winLoss << endl;
678     cout << setw(rowLength) << tableRow << endl;
679 
680     for (int i = 0; i < *(gameSettings + 2); i++)
681     {
682         cout << right << setw(maxSpace) << playerInfo[i].fullName << " :" <<      ↵
              right << setw(5)
683             << playerInfo[i].stats[0] << " :" << right << setw(5) << playerInfo  ↵
                [i].stats[1]
684             << " :" << right << setw(5) << playerInfo[i].stats[2] << " :" <<     ↵
                endl;
685         cout << setw(rowLength) << tableRow << endl;
686     }
687 
688     cout << endl;
689 }
```