#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SIZE 10 /\* constant for size of game board \*/

/\* function prototypes \*/

void printStudInfo(void);

void printIntro(int maxTries);

void printGridRow(int i);

void dispGrid(int \*\*board);

int assignShip(int \*\*board, int shipType);

int main(void)

{

int \*\*board; /\* pointer to two-dimensional array for game board \*/

int i; /\* index used in loop counters, specifically outer loop \*/

int j; /\* index used in loop counters, specifically inner loop \*/

char level[10] = ""; /\* user input for difficulty of game \*/

int maxTries; /\* number of shots available to user based on difficulty level \*/

int validLoc = 0; /\* boolean indicating whether or not valid location found to assign ship \*/

printStudInfo(); /\* display student information \*/

printf("\n Welcome to Battleship! \n\n");

/\* prompt user to enter difficulty level of game \*/

while ((strcmp(level, "easy")) && (strcmp(level, "normal")) && (strcmp(level, "hard")))

{

printf("Enter difficulty level of game (easy, normal, hard): ");

scanf("%s", level);

if ((strcmp(level, "easy")) && (strcmp(level, "normal")) && (strcmp(level, "hard")))

{

printf("Invalid input. Response is case sensitive. Please select again.\n");

}

}

/\* assign number of shots available based on difficulty level chosen \*/

if (!strcmp(level, "easy"))

{

maxTries = 30;

}

else if (!strcmp(level, "normal"))

{

maxTries = 25;

}

else /\* strcmp(level, "hard")) \*/

{

maxTries = 20;

}

printIntro(maxTries); /\* call function to print introductory message, including number of shots \*/

printf("Initializing board... ");

/\* allocate memory for and initialize two-dimensional board \*/

board = calloc(SIZE, sizeof(int \*));

for(i = 0; i < SIZE; i++)

{

board[i] = calloc(SIZE, sizeof(int));

}

/\* now place battleship on board \*/

printf("now let's begin!\n\n");

/\* repeatedly attempt to assign battleship until successful \*/

while (!validLoc)

{

validLoc = assignShip(board, 4);

}

validLoc = 0; /\* re-initialize valid location to false for aircraft carrier \*/

/\* repeatedly attempt to assign aircraft carrier until successful \*/

while (!validLoc)

{

validLoc = assignShip(board, 5);

}

dispGrid(board); /\* call function to display board with ships assigned \*/

return 0;

}

/\*

============================================================================

Function : printStudInfo

Parameters : none

Return : none

Description : This function prints the student header information. ============================================================================

\*/

void printStudInfo(void)

{

/\* display student information \*/

printf("+----------------------------------------------+\n");

printf("| Computer Science and Engineering |\n");

printf("| CSCE |\n");

printf("| Student Name |\n");

printf("+----------------------------------------------+\n\n");

}

/\*

============================================================================

Function : printIntro

Parameters : integer representing number of shot available to player based

on level of difficulty

Return : none

Description : This function prints an introductory message to the user, and

gives details about the game including the number of attempts

the user gets based on their chosen level of difficulty.

============================================================================

\*/

void printIntro(int maxTries)

{

printf("\n");

printf("----------------------------------------------------------\n");

printf("The computer program will randomly assign an aircraft car-\n");

printf("rier and a battleship, that are oriented either vertically\n");

printf("or horizontally, to the board. You will have %d chances to\n", maxTries);

printf("sink both the computer's aircraft carrier and battleship!!\n");

printf("You'll play on a %d x %d board, where the aircraft carrier\n", SIZE, SIZE);

printf("will have a length of 5 units and the battleship will have\n");

printf("a length of 4 units. \n");

printf("----------------------------------------------------------\n\n");

}

/\*

============================================================================

Function : printGridRow

Parameters : integer representing row of the board

Return : none

Description : This function prints the corresponding letter for the integer

row.

============================================================================

\*/

void printGridRow(int i)

{

/\* based on row integer passed, print out corresponding row letter of board \*/

switch (i)

{

case 0: printf("A | ");

break;

case 1: printf("B | ");

break;

case 2: printf("C | ");

break;

case 3: printf("D | ");

break;

case 4: printf("E | ");

break;

case 5: printf("F | ");

break;

case 6: printf("G | ");

break;

case 7: printf("H | ");

break;

case 8: printf("I | ");

break;

case 9: printf("J | ");

break;

default:printf("Unknown grid row value: %d. Program terminating.\n", i);

exit(1);

}

}

/\*

============================================================================

Function : dispGrid

Parameters : pointer to two-dimensional array representing game board

Return : none

Description : This function prints the game board, including row and column

headers and assigned ships.

============================================================================

\*/

void dispGrid(int \*\*board)

{

int i; /\* index used in outer loop counter \*/

int j; /\* index used in inner loop counter \*/

printf(" 1 2 3 4 5 6 7 8 9 10 \n");

printf(" +---------------------+\n");

/\* loop through two-dimensional array to print out corresponding items \*/

for (i = 0; i < SIZE; i++)

{

printGridRow(i); /\* print grid row and left vertical column of board \*/

for (j = 0; j < SIZE; j++)

{

switch (board[i][j])

{

case 0: /\* print blank space for open/unassigned location \*/

printf(" ");

break;

case 4: /\* print 'B' for assigned battleship \*/

printf("B ");

break;

case 5: /\* print 'A' for assigned aircraft carrier \*/

printf("A ");

break;

default:/\* should never get here, but handle error case by terminating program gracefully \*/

printf("Unknown value for board[%d][%d] : %d. Terminating program.\n", i, j, board[i][j]);

exit(1);

}

}

printf("|\n"); /\* print right vertical column of board \*/

}

printf(" +---------------------+\n");

}

/\*

============================================================================

Function : assignShip

Parameters : pointer to two-dimensional array representing game board, and

integer representing ship type, based on the length

Return : integer, 0 if ship not assigned or 1 if ship successfully as-

signed

Description : This function attempts to randomly assign the requested ship,

either an aircraft carrier or a battleship, to the board.

============================================================================

\*/

int assignShip(int \*\*board, int shipType)

{

int valid = 0; /\* use 0 for false (not assigned); otherwise 1 means assigned successfully \*/

int i; /\* index used in loop counter as row \*/

int j; /\* index used in loop counter as column \*/

int vertical; /\* randomly generated ship orientation \*/

int shipRowPos; /\* randomly generated row position of ship \*/

int shipColPos; /\* randomly generated column position of ship \*/

srand(time(NULL));

vertical = rand() % 2; /\* randomly generated ship orientation: 1 = vertical, 0 = horizontal \*/

shipRowPos = rand() % 10; /\* randomly generated ship row position \*/

shipColPos = rand() % 10; /\* randomly generated ship column position \*/

if (vertical) /\* vertical orientation \*/

{

/\* check to see if vertical ship can be placed on board at this location \*/

if ((shipRowPos + shipType) < SIZE)

{

/\* need check if ship already in this position first \*/

for (i = shipRowPos; i < (shipRowPos + shipType); i++)

{

if (board[i][shipColPos] != 0)

{

/\* ship already here, return valid = 0 \*/

return valid;

}

}

/\* ship able to fit vertically on board \*/

valid = 1;

/\* need assign ship's position on board \*/

for (i = shipRowPos; i < (shipRowPos + shipType); i++)

{

board[i][shipColPos] = shipType;

}

}

}

else /\* horizontal orientation \*/

{

/\* check to see if horizontal ship can be placed on board at this location \*/

if ((shipColPos + shipType) < SIZE)

{

/\* need check if ship already in this position first \*/

for (j = shipColPos; j < (shipColPos + shipType); j++)

{

if (board[shipRowPos][j] != 0)

{

/\* ship already here \*/

return valid;

}

}

/\* ship able to fit vertically on board \*/

valid = 1;

/\* need assign ship's position on board \*/

for (j = shipColPos; j < (shipColPos + shipType); j++)

{

board[shipRowPos][j] = shipType;

}

}

}

return valid;

}