Array Operations

- Run the provided solution at

  /home/work/cpe112/Executables/Project_11/Project_11_solution
- Run the comparison script at

  /home/work/cpe112data/Project_11/CompareSolution.bash Project_11.cpp
- Note the output format of any messages
  - Run all input files provided to see the results
  - Informational/Error messages occur due to
    - Invalid input file name
    - Empty input file
    - Invalid row and/or column index
    - Repeat of a valid row – column index pair

Program Requirements

- Use a 10X10 character array
  - Initialize every element to a period '.'
  - Boats are placed into the array using characters 1, 2, 3 or 4
- Files that contain the header line will not have any missing information
- All inputs read from standard input (cin) must be echo printed
- Solution must contain at least 3 functions not including main

Input File

- **Open the input file in a loop that continues to execute until a valid filename is entered or ctrl-c is used to terminate the program**
- **Input file Contents (comments not present):**

```
Row     Col     Boat       // header line of the file
1       1       1
2       4       2
1       7       3          // 12 lines of row and col index
4       8       4          // values for boat placement
...
```


Reading Input File

- Use the getline function to read the header line from the input file
  - If the read is successful, then the file contains the rest of the information necessary for successfully placing the 12 boat positions in the character array
  - If reading the header is not successful, then the input file is empty.
- Read in the row and column value for a boat
  - Row and Column values read as integers – use extraction
- Read in the boat number as a character
  - Use the extraction operation to read the boat into a character variable
- Place the boat number in the array at the indicated row, column index value

Checking User Selection

- For a row-column pair provided by the user
  - Verify that the row and column indexes are valid
    - Invalid row/column indexes result in an error message
  - Verify that the position specified has not been selected previously
    - Previously selected positions result in an error message
  - Update the board showing the position selected as being a hit or a miss
    - use an H for a hit
    - use an X for a miss

Miscellaneous output

- Output hit or miss for valid first time selections
- When a boat has been sunk, all three boat positions have been hit, print out that the boat has been sunk
  - Use static variables for each boat to keep track of the number of hits for that boat
- Once 12 hits have been recorded, all boats have been sunk and the game is over.
  - Print out the congratulations
  - Print out the successful hit percentage
  - Print out the final board

Possible Algorithm for main

- Open the input file (function call)
- Read the header to see if the file is empty
- Initialize the game board (function call)
- Place the boats (function call)
- Loop while total number of hits is < 12
  - Obtain a valid move (function call)
  - Check the valid move (function call)
  - If the move was a hit, increment total number of hits
  - Increment total number of shots taken
- Game is over, print out message at the end of the game.

Possible Algorithm for Obtain Move

- Start a do-while loop
  - Print out the game board
  - Initialize continue to loop flag to false
  - Prompt for and read in a row and column index value
  - Test for invalid row, invalid column or invalid row and column index values. If invalid change flag to true
  - If index values are valid, verify that the spot has not been previously selected – board does not contain an X or an H at that position. If previously selected, change flag to true
  - Continue to loop until index values are valid and the spot has not been previously selected: flag has a value of true
- Use a flag controlled loop to control this loop.