More arrays:
Another parallel array example:
To play hangman have an array w of six characters for the input word and a parallel array cor of zeros or ones for whether they have guessed the correct letter for a location.
E.G. Player one inputs the word “papers”. Then
w[0]=’p’,w[1]=’a’,w[2]=’p’,w[3]=’e’,w[4]=’r’,w[5]=’s’.
The screen is cleared by using a for loop, and then player two gets to open their eyes and guesses. Say they guess ‘e’. Then the program prints out “e in fourth place” and cor[3] is changed from 0 to 1. Suppose they guess ‘p’. Then the program prints out “p in first and third place” and cor[0] and cor[2] are changed from 0 to 1. The program ends when they have made seven incorrect guesses or have correctly guessed the word. We will look at this example in more depth when we are studying functions.

Two-dimensional arrays (or matrices)
Basic:
1) Read in or print out a two dimensional array
2) Compute the sums of the rows and print these out.
3) Compute the average of the sums of the rows.
4) Compute the sum of the main diagonal elements
5) Add two matrices together.
6) See if two matrices are equal.
7) Multiply a matrix by a number.
8) See if a matrix is symmetric ($a_{ij} = a_{ji}$)
Less basic:
1) Use a two dimensional array to keep track of the scores of two player teams:

2) Use a two dimensional array for creating designs. E.g.

```
char a[6][6]={‘*’,’* ‘,’*’,’* ‘,’*’,’* ‘,
              ‘ ‘,’*’,’ ‘,’ ‘,’ ‘,’ ‘,
               ’,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
              ’,‘,’ ‘,’*’,’ ‘,’ ‘,’ ‘,
               ’,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
              ’,‘,’ ‘,’ ‘,’*’,’ ‘,’ ‘,
               ’,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
              ’,‘,’ ‘,’ ‘,’ ‘,’*’,’ ‘,
               ’,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
               ‘*’,’* ‘,’*’,’* ‘,’*’,’* ‘}
```

3) Use a three dimensional array, declared as char tic[3][3][3], to write a computer game to play three dimensional tic-tac-toe.