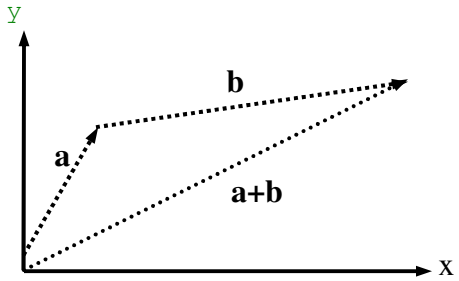
CFJ PART 1 – BASIC C/C++

Exercise Sheet 2 – Building up a head of steam (Arrays, Strings & Functions)

1. We can use an array with two slots, each containing a number represented as a double type, to denote the mathematical idea of a *vector* in 2-dimensional space. Implement a *function* that performs vector addition on two **global** arrays, **a** and **b**, and places the results in a third global array, **c**. i.e. **c = a + b**, where **c[0] = a[0] + b[0]** and so on

A vector, $v \equiv \begin{pmatrix} x \\ y \end{pmatrix}$ has corresponding array representation: **double v[2] = {x, y}**



2. Write a program that prompts the user to type in a list of integers until 0 is entered. Store these values, as they are input, into an array called ***userInput***, and when the user has completed entering data, use a for loop to display all of these values to screen.

Then add a segment of code to your main program that reverses the order of the entries inside of the array before they are displayed (you may want to construct a temporary array to help you do this).

3. Add a function to the program in the last exercise called ***display***, that ouputs to screen the integer passed to it. followed by an appropriate number of stars (creating a sort of histogram). For example if the user inputs were 5, 9, 4, 12 and 2 then output would be:

```
5: *****
9: *********
4: ****
2: **
7: *******
```

4. Remember the factorial exercise in the last exercise sheet? Re-write that exercise but this time using a **recursive** function.

5. Functions in the Standard Libraries are nothing mystical – they're just code written by other people to make your life easier. If it came to it you could easily write your own versions of most of these functions.

To prove this to yourself, write your own version of a function that works out the number of characters in a null terminated string, called **strlen**, and test it out. The prototype for your function should be of the form:

```
int strlen(const char s[])…
```

6. Write the body of a *function* called **reverse** that takes in an array of characters as input and returns them in another array in reverse order so you can display them to screen. For example the code in your main function would be something like:

```
const int N = 11;
input[N] = "doG si nevS";
output[N] = "";
reverse(input, output, N);
cout << output << endl;
```

7. Use the fact that our null terminated strings are just char arrays - and hence allow us to use standard array indexing - to write a function called **isPalindrome** that returns a boolean to your main program. (A palindrome is a word that reads the same backwards as it does forwards)

You could write the code for this from scratch – but why not utilise the reverse function that you have written for the last exercise to create a reversed copy of the string, and then just compare the two. It is fine to have functions calling other functions.

8. Write a noughts and crosses game for two human players. This should encompass all of the areas we are covering in this exercise sheet:

- The game board should be stored in memory as a 2-dimensional array.
- Use a display function to ouput the graphical representation of this onto screen.
- Use **`cin`** to input a null terminated string, indicating where the next player wants to place their move.
- Use functions to analyse this string, and so deduce how to update the game board.
- Have a function that checks to see whether anyone has won.

Try and include some error checking to determine whether a move is valid or not, and to make sure that the program won't just break down in tears if someone enters the wrong format for a move. If you find this all really easy, you could even add some AI for a computer opponent….