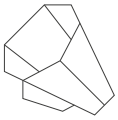

Read this first!!

When doing the exercises, it is advised that you make notes! These notes will be useful for the assignment you have to write later on in the course.

Exercise 4.1 my_strdiff, my_strlen, my_strcpy, my_strdup

1. Implement a function `my_strdiff` that compares two strings (`char* a, char* b`) and returns -1 if the two strings are identical or the first position that they differ. You **may not** use built-in library functions to implement this function.
2. Make three tests of the function, using constant strings as parameters:
   - Compare "Hello World" with "Hello World"
   - Compare "Hello World" with the empty string
   - Compare two empty strings
   - Compare "Hello World" with "Hello, world"
3. Implement your own versions of the functions `strlen`, `strcpy`, and `strdup` (be inspired by Kernighan and Ritchie). You may not use built-in functions to implement your own functions.
4. Make tests of the functions.

Exercise 4.2 Linked List

Design (<u>on paper</u>) the structure of your linked list; describe how an element is added and removed and. Design for all cases: empty list, half-full list etc.

Implement a linked list in two files list.h and list.c

The items to be stored in the list is a given as a void pointer (`void *`) that points to the element. In this way, you can implement a generic linked list that will be able to hold any kind of elements.

list.h must at least have this contents:

```
int add_item(void * item); // Return 0 if item added else -1
void *get_item(uint16_t index); // Return pointer to item at given index in the list
int no_of_items(); // Return no of items in list
int remove_item(void * item); // Return 0 if item removed else -1
```

1. Implement the functions in list.c using Test Driven Development
2. Write a small main() to demonstrate your linked list.