Assignment 7

Exercise 00: vc_strdup

Turn-in files	**vc_strdup.c**
Allowed functions	malloc

- Reproduce the behavior of the function **strdup** (`man strdup`)

- Here's how it should be prototyped:

```
char *vc_strdup(char *src);
```

Exercise 01: vc_range

Turn-in files	**vc_range.c**
Allowed functions	malloc

- Create a function that returns an array of **int**s. This **int** array should contain all values between **min** and **max**.
- **min** included - **max** excluded.
- Here's how it should be prototyped:

```
int *vc_range(int min, int max);
```

- If **min** value is greater or equal to **max**'s value, a null pointer should be returned.

Exercise 02: vc_ultimate_range

Turn-in files	**vc_ultimate_range.c**
Allowed functions	malloc

- Create a function that allocates and assigns an array of **int**s. This **int** array should contain all values between **min** and **max**.
- **min** included - **max** excluded.
- Here's how it should be prototyped:

```
int vc_ultimate_range(int **range, int min, int max);
```

- If **min** value is greater or equal to **max**'s value, **range** will point on *NULL*.

- The size of **range** should be returned (or 0 on error).

Exercise 03: vc_concat_params

Turn-in files	vc_concat_params.c
Allowed functions	malloc

- Create a function that transforms arguments given as command-line into a single string of characters. Those arguments should be separated by a **"\n"**
- Here's how it should be prototyped:

```
char *vc_concat_params(int argc, char **argv);
```

Exercise 04: vc_split_whitespaces

Turn-in files	vc_split_whitespaces.c
Allowed functions	malloc

- Create a function that splits a string of characters into words.
- Separators are spaces, tabs and line breaks.
- This function returns an array where each box contains a character-string's address represended by a word. The last element of this array should be equal to 0 to emphasize the end of the array.
- There can't be any empty strings in your array. Draw the necessary conclusions.
- The given string can't be modified.
- Here's how it should be prototyped:

```
char **vc_split_whitespaces(char *str);
```

Exercise 05: vc_print_words

Turn-in files	vc_print_words.c
Allowed functions	putchar

- Create a function that displays the content of the array you created in the last exercise's function (Exercise 04).
- One word per line.
- Each word will be followed by a **"\n"**, including the last one.
- This exercise will be compiled with your vc_split_whitespaces.c for testing.
- Watch out not to have multiple **#define**
- Here's how it should be prototyped:

```
void vc_print_words(char **words);
```

Exercise 06: vc_convert_base

Turn-in files	**vc_convert_base.c**
Allowed functions	malloc, free

- Create a function that returns the result of conversion of the string **nbr** from a base **base_from** to a base **base_to**. The string must have enough allocated memory. The number represented by **nbr** must fit inside an **int**.
- Example: "17" base_from 10 base_to 16, should return "11".
- Here's how it should be prototyped:

```
char *vc_convert_base(char *nbr, char *base_from, char *base_to);
```

Exercise 07: vc_split

Turn-in files	**vc_split.c**
Allowed functions	malloc

- Create a function that splits a string of character depending on another string of characters.
- You will have to use each character from the string **charset** as a separator.
- The function returns an array where each box contains the address of a string wrapped between two separators. The last element of that array should equal to 0 to indicate the end of the array.
- There cannot be any empty strings in your array. Draw your conclusions accordingly.
- The string given as argument won't be modifiable.
- Here's how it should be prototyped:

```
char **vc_split(char *str, char *charset);
```