Problem Sheet #1

Problem 1.1: *freshie crash*

The program is producing unexpected results because of the *strdup* function.
The function returns *strncpy*, which returns a pointer to the destination string. In this case that is a pointer to *d*. However, *d* is a local variable and after the function returns there is no existence of *d* - so the pointer may contain garbage values. The use of said returned pointer leads to undefined behavior - and unexpected results.

Problem 1.2: *memory segments*

```c
#include <stdlib.h>
#include <stdio.h>
#include <stdio.h>

char *strdup(const char *s) // Stack Segment
{
    char *p = NULL; // Stack Segment
    size_t len; // Stack Segment

    if (s) {
        len = strlen(s);
        p = malloc(len+1); // Heap Segment
        if (p) {
            strcpy(p, s);
        }
    }
    return p; // Stack segment
}

int main() // Stack Segment
{
    static char m[] = "Hello World!"; // Data Segment
    char *p = strdup(m); // Stack Segment
    if (!p) {
        perror("strdup");
        return EXIT_FAILURE; // Stack Segment
    }
    if (puts(p) == EOF) {
        perror("puts");
        return EXIT_FAILURE; // Stack Segment
    }
    if (fflush(stdout) == EOF) {
        perror("fflush");
        return EXIT_FAILURE; // Stack Segment
    }
    return EXIT_SUCCESS; // Stack Segment
}
```

The text segment stores the machine instructions of the program.

The static variable `m[]` and `"Hello World!"` is stored in the data segment.

The heap segment stores the buffer from `malloc` pointed to by `p`.

The stack stores return addresses of the function. Also, it stores the parameters of functions - so `const char *s` is stored in the stack segment. Local variables like `len` in the `strdup` function and `char *p` in the `main` function is stored in the stack segment. Similarly, management of the data required by the function calls made in both `main` and `strdup` like `strlen`, `puts` and `fflush` are stored in the stack segment.

Problem 1.3: *execute a command in a modified environment or print the environment*

Please find the source code `env.c`