Question #1

```
#include <stdio.h>

int main() {
    int array[4] = {1, 4, 7, 10};
    int *ptr1= (int *)(&array + 1);
    printf("%d\n", ptr1 - array);
    int *ptr2= (int *)(array + 1);
    printf("%d\n", ptr2 - array);

    return 0;
}
```

Comparing by value, `&array` and `array` both hold the same values: `&array[0]` but considering types, they're different:

`array` is `int*`

`&array` is `int (*)[16]`

And this difference in type, makes these objects behave differently when doing mathematical operations such as + and -.

`array + 1` is the address of `array[1]`, but

`&array + 1` is the address of first byte after last item of `array`.

Question #2

```
#include <stdio.h>


int main() {
    ~~int x[0], a=2;~~
    // 1 - Size of an array should be positive
    int x[1], a=2;


    ~~int *b = a, *x_ptr = &x;~~
    // 2 - value of pointer should be an address (*b = &a).
    // 2.5 - *x_ptr = &x, would do the job too, since they both
hold the same value, but they conflict in type.
    int *b = &a, *x_ptr = x;


    scanf("%d %d", b, x_ptr);


    ~~printf("a=%d, x=[%d]\n", *b, *x[0]);~~
    // 3 - x[0] is not a pointer (nor address) so we cannot use
*x[0] syntax


    printf("a=%d, x=[%d]\n", *b, x[0]);


    return 0;
}
```

The program takes two values from user and stores them in a and `x[0]`.

Question #3

```c
#include <stdio.h>

int main() {
    int array[100] = {};
    // I'm pretty sure ; is a typo mistake here, isn't it?
    for (int i=0; i<100; i++)
        array[i] = 3*i + 1;

    printf("%d\n", array);
    printf("%d\n", array[5]+1);
    printf("%d\n", &array[5]+1);

    return 0;
}
```

`array` holds the value `&array[0]` which is `x1000 =>` `4096`

`array[5] + 1 = 3*5 + 1 + 1 = 17 =>` `17`

`&array[5] + 1 = &array[0] + 5*4` (bytes) `+ 1*4 = 100 =>` `4120`

Question #4

```
#include <stdio.h>

int main() {
    FILE* my_file = fopen("file.txt", "w");

    fputs("Help", my_file);
    fseek(my_file, 3, SEEK_SET);
    fputs("local", my_file);
    fseek(my_file, 5, SEEK_SET);
    fputs("Friday", my_file);
    fseek(my_file, 8, SEEK_SET);
    fputs("end!", my_file);

    return 0;
}
```

1) We put `"Help"`.

2) We move the pointer to after `l` in `"Helo"`.

3) We put `"local"` => text is `"Hellocal"`.

4) We move the pointer to after o in `"Hellocal"`.

5) We put `"Friday"` => text is `"HelloFriday"`.

6) We move the pointer to after `i` in `"HelloFriday"`.

7) We put `"end!"`, so the text is going to be `"HelloFriend!"`!