Homework: C Pointers

This document defines homework assignments from the "C Programming" Course @ Software University. Please submit as homework a single **zip** / **rar** / **7z** archive holding the solutions (source code only) of all below described problems.

Problem 1. Swap numbers

Implement a function which takes as input **two numbers** and **swaps their values**. The declaration of the function should be something like:

```
void swap(int *first, int *second)
```

Problem 2. Array Print

Declare an array of integers and print it on the console. Do NOT use the indexer operator **[]**.

Problem 3. Print Array Reversed

You are given a sample array as input and your task is to print the array in reversed order using **pointer arithmetic** instead of indexing.

Input	Output
4 1 2 3 4	4 3 2 1
5 5 8 2 9 10	10 9 2 8 5

Problem 4. Print Integer Address

Write a function which takes as input an integer and prints the address of it in the memory.

Try printing the address of the integer in the function and in the main function. What is the difference and why?

Input	Output
print(5)	0x2686780 (The output varies depending on the address on your computer)

Problem 5. Create New Integer

Write a function that declares and initializes an integer on the stack, then returning it. Try creating the function with two different declarations:

```
int new_integer()
```

```
int* new_integer_ptr()
```

What is the difference between the two declarations? Why is the second declaration **not safe** to use? Give an example.

Problem 6. Digit Hate

Write a function that takes a **string as input** and **replaces all occurrences of a digit** (0-9) with a slash **'/'**.

The function should **NOT modify** the original string. It should **return the resulting string** as result. The function should also notify the calling function how many digits were replaced. Find a way to do this using pointer parameters.

Tip: Be careful where you allocate the resulting string.

Input	Output
`The Cold War is believed to be between the period 1947-1999.`	`The Cold War is believed to be between the period ////-////.`

Problem 7. Generic Memory Dump Function

Write a function that takes a **pointer of any type,** size of bytes **n** and row length **len**. The function should print a total of **n** bytes, starting from the address of the pointer.

The output should be formatted into several rows, each holding **len** bytes. At the start of each row, print the address of the initial byte.

Note: The addresses may vary.

Input	Output

`char *text = "I love to break free";`
`mem_dump(text, strlen(text) + 1, 5);`
	`0x400742 49 20 6c 6f 76`
	`0x400747 65 20 74 6f 20`
	`0x40074c 62 72 65 61 6b`
	`0x400751 20 66 72 65 65`
	`0x400756 00`

Input	Output

`int array[] = { 7, 3, 2, 10, -5 };`
`size_t size = sizeof(array) / sizeof(int);`
`mem_dump(array, size * sizeof(int), 4);`
	`0x7fff48bb8a90 07 00 00 00`
	`0x7fff48bb8a94 03 00 00 00`
	`0x7fff48bb8a98 02 00 00 00`
	`0x7fff48bb8a9c 0a 00 00 00`
	`0x7fff48bb8aa0 fb ff ff ff`

Problem 8. Generic Swap Function

Write a function that takes **2 pointers** and **swaps the memory** they point to. The function should also take **the size of each memory piece** and should work with any data type.

Sample Code	Output

`char letter = 'B', symbol = '+';`
`swap(&letter, &symbol, sizeof(char));`
`printf("%c %c", letter, symbol);`
	`+ B`

Sample Code	Output

`int a = 10, b = 6;`
`swap(&a, &b, sizeof(int));`
`printf("%d %d", a, b);`
	`6 10`

Sample Code	Output

`double d = 3.14, f = 1.23567;`
`swap(&d, &f, sizeof(double));`
`printf("%.2f %.2f", d, f);`
	`1.23 3.14`

Problem 9. Clients

Declare a **struct** called **Client** with the following members: **name**, **age** and **account balance**. More on structures: http://www.tutorialspoint.com/cprogramming/c_structures.htm

Write a function that sorts an array of clients using a specific **comparator**. The comparator should be another function that **determines how the clients are sorted**. **Write comparators** for sorting the clients by name, age and account balance.

Sample Code

`struct Client clients[] = { ... };`
`int clientsCount = ...;`
`sort_clients(clients, clientsCount, &name_comparator);`
`sort_clients(clients, clientsCount, &balance_comparator);`
`sort_clients(clients, clientsCount, &age_comparator);`

Problem 10. ** Line Reading Function

Write a function that **reads an entire line** from the standard input stream (console) and returns a pointer to the **read string**. The function should be able to read lines of **unknown size**.

Hint: Initially allocate a **small buffer** (use `malloc()`) and **increase** its size when it gets full (use `realloc()`).

Forbidden: Do not use hacks such as declaring very large buffers.