Pointer Arithmetic, pointer dereferencing (Solutions)

This problem tests your understanding of casting and pointer de-referencing. Consider the following code, being executed on a Little Endian Pentium, 32-bit machine where

\[
\begin{align*}
\text{sizeof(char)} & \quad = 1 \\
\text{sizeof(short)} & \quad = 2 \\
\text{sizeof(int)} & \quad = 4 \\
\text{sizeof(int *)} & \quad = 4
\end{align*}
\]

The size of any pointer (e.g. char *) is 4 bytes.

For each of the following assignment statements, fill in the blanks in the comments to indicate the result of the assignment. All answers must be in hex.

```c
int main()
{
    int array[3];
    int * ptr;
    int x;

    array[0] = 0xaabbccdd;
    array[1] = 0x55667788;
    array[2] = 0x11223344;
    ptr = array;

    x = *(int *)((int *)ptr + 1); /* x = 0x55667788 */

    x = *(int *)((char *)ptr + 1); /* x = 0x88aabbcc */

    x = *(int *)((char **)ptr + 1); /* x = 0x55667788 */

    x = *(int *)((long *)ptr + 1); /* x = 0x55667788 */

    x = *(int *)((short *)ptr + 1); /* x = 0x7788aabb */
}
```
Now, consider the following code, being executed on a Little Endian Pentium, 64-bit machine where,

```
sizeof(char) == 1
sizeof(short) == 2
sizeof(int) == 4
sizeof(int *) == 8
```

The size of any pointer (e.g. char *) is 8 bytes.

For each of the following assignment statements, fill in the blanks in the comments to indicate the result of the assignment. All answers must be in hex.

```c
int main()
{
    int array[3];
    int * ptr;
    int x;

    array[0] = 0xaabbccdd;
    array[1] = 0x55667788;
    array[2] = 0x11223344;
    ptr = array;

    x = *(int *)((int *)ptr + 1);
    /* x = 0x55667788 */

    x = *(int *)((char *)ptr + 1);
    /* x = 0x88aabbcc */

    x = *(int *)((char **)ptr + 1);
    /* x = 0x11223344 */

    x = *(int *)((long *)ptr + 1);
    /* x = 0x11223344 */

    x = *(int *)((short *)ptr + 1);
    /* x = 0x7788aabb */
}
```