Colby Frison
OUID: 113568816

CS-3323 || Homework 8

4/29/2025

8.1 Suppose we are compiling for a machine with 1-byte characters, 2-byte shorts, 4-byte integers, and 8-byte reals, and with alignment rules that require the address of every primitive data element to be an even multiple of the element's size. Suppose further that the compiler is not permitted to reorder fields. How much space will be consumed by the following array? Explain.

```
A : array [0..9] of record
      s : short  → 2 bytes
      c : char   → 1 byte
      t : short  → 2 bytes
      d : char   → 1 byte
      r : real   → 8 bytes
      i : integer → 4 bytes
```

Each data element must be aligned to a memory address that is a multiple of its size.

Array	size/alignment	padding	offset
s short	2	0	0
c char	1	0	2
(padding)		1	3
t short	2	0	4
d char	1	0	6
(padding)		1	7
r real	8	0	8
i int	4	0	16
(padding)		4	20

→ total of 24

since the array has 10 records $10 \cdot 24 = 240$ bytes

8.12 Consider the following C declaration, compiled on a 64-bit x86 machine:

```
struct {
    int n;
    char c;
} A[10][10];
```

If the address of `A[0][0]` is 1000 (decimal), what is the address of `A[3][7]`?

What is offset of A[3][7]

offset = $(i \times 10 + j) \times 8$

$(3 \times 10 + 7) \times 8 = 296$

Shift ⇒ $1000 + 296 = 1296$

(1000 = A[0][0])

8.16 Explain the meaning of the following C declarations:

```
double *a[n];
double (*b)[n];
double (*c[n])();
double (*d())[n];
```

Final answers

1. is simple
2. b is a pointer that points to an array of n elements, where each element is a double
3. c is an array of pointers to a function that takes unspecified parameters which returns a double
4. d is a function that takes no parameters and returns a pointer to an array of doubles