Problem ST-4 (3 parts) Pointers, Arrays, and Structs

Assuming a 32-bit system with 32-bit memory interface and 32-bit addresses, answer the following questions.

Part A Show how the `struct` definition below maps to memory. Assume it is allocated starting at address 5000. For each variable, draw a box showing its size and position in memory. Label the box with the variable name. Label each element of an array (e.g., Name[0]).

```
struct Dog {
   char          Name[6];
   unsigned char Age;
   float         Weight;
   struct Breed  *MyBreed;
   struct Dog    *Next;
}
```

5000	*Name[0]*	*Name[1]*	*Name[2]*	*Name[3]*
5004	*Name[4]*	*Name[5]*	*Name[6]*	*Age*
5008	*Weight*			
5012	*MyBreed*			
5016	*Next*			
5020				
5024				

Part B Suppose the following variables are allocated beginning at address 6000. Complete the table below, listing the value of the expression following this definition.

```
int         A[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
int         *P = A;
```

A	**6000**	**A + 5**	**6020**	**&(A[9])**	**6036**	**P - 1**	**5996**
A[9]	**9**	**P[5]**	**5**	**A == P**	**1**	**P++**	**6004**

Explain what happens if **A** is incremented (e.g. **A++**).

The compiler will issue an error stating A is a constant, not a variable; therefore it cannot be incremented.

Part C Write the MIPS code implementation of the dynamically allocated array access below in the smallest number of instructions. A pointer to the array (declared below) is stored in $3. Variables X, Y, Z, and R reside in $4, $5 $6, and $2, respectively. Modify only $1 and $2.

```
int      Array[64][16][32];     /* array declaration */
R = Array[Z][Y][X];             /* implement this */
```

Label	Instruction	Comment
	sll $1, $6, 9	*# 512 * Z*
	sll $2, $5, 5	*# 32 * Y*
	add $1, $1, $2	*# offset = 512Z + 32Y*
	add $1, $1, $4	*# offset += X*
	sll $1, $1, 2	*# byte offset = 4 * offset*
	add $1, $3, $1	*# ptr = Array + byte offset*
	lw $2, 0($1)	*# R = *ptr*