Structures in One Shot

What are Structures?

→ User defined data types

S

```
struct employee{
      int emp_id;
      char name;
      float salary;
};
```

4 bytes

int x ; →

float y;

Char ch ; → 1 byte

Class → students → roll no, marks → phy, chem, marks

Rohan → 874 , "Rohan" , 80,000.78

Why Structures?

Raghav

Sanket

Manvi

Urvi

Grade, Percentage, Roll No

Grade → char

Percentage → float

Roll No → int

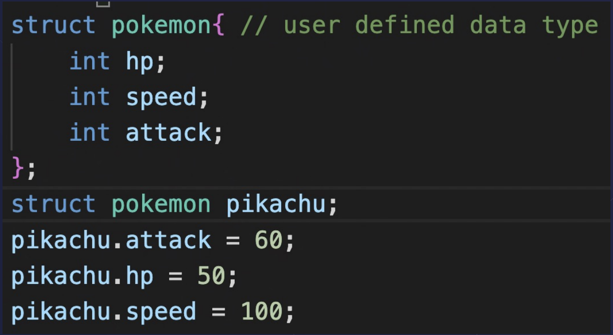
char grade[]	'B'	'A'	'B'	'A'
float per[]	90	85.5	74.3	98.1
int roll[]	45	56	81	92

Pokemon's

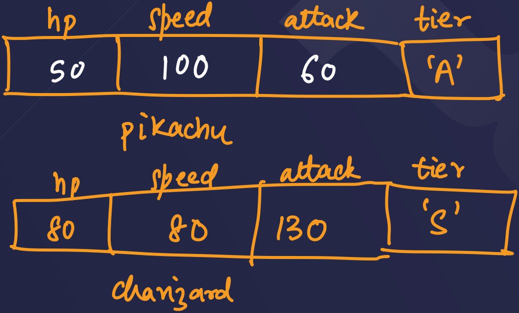
Pikachu	60 Health	100 Speed	70 attack
Charizard	80 Health	80 Speed	130 attack

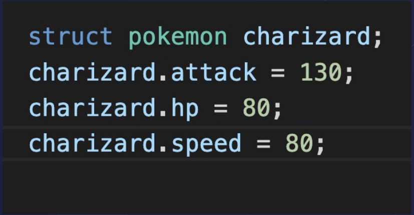
Laptop → Processer, Storage, Ram, Screen, keyboard, trackpad

```
int x;
int y;
```



```
struct pokemon{ // user defined data type
    int hp;
    int speed;
    int attack;
};
struct pokemon pikachu;
pikachu.attack = 60;
pikachu.hp = 50;
pikachu.speed = 100;
```





```
struct pokemon charizard;
charizard.attack = 130;
charizard.hp = 80;
charizard.speed = 80;
```

Array approach vs Structures

50 car → name

Multiple attributes → diff types

Car → name, Engine Power, size, no. of seats

Ques : What should be preferred to store 10 floats in a memory? Array or structure?

→ Array, 'same data type'

Ques : State true or false

An array should be used to store dissimilar elements, and a structure to store similar elements.

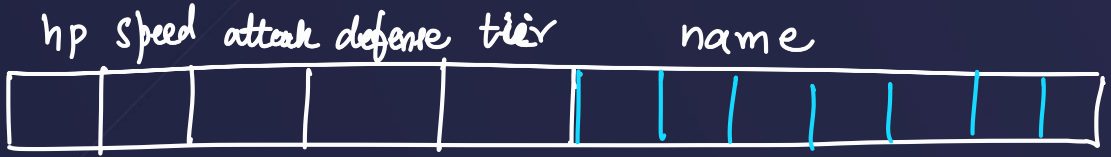
→ False

Structure variables

Declaration, Initialization and Accessing

```
struct~~ure~~ pokemon {
    int hp;
    int speed;
    int attack;
    int defense;
    char tier;
    char name[15];
}
```

```
struct pokemon pikachu;
```



dot operator :

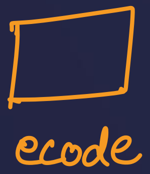
```
pikachu . attack = 60;
```

Ques : Create a structure type ‘book’ with name, price and number of pages as its attributes

```
Struct  book {
        char  name [50];
        float  price;
        int  noOfPages;
} .
```

Ques : Find the error

```
Struct emp{
      int ecode;
      struct emp e;
};
```



```
struct emp {
      int ecode;
};

struct emp e;
```

Objects & Classes

```
struct pokemon{
        int hp;
        int attack;
        int speed;

}
struct legendaryPokemon{
        int specialattack;
        struct pokemon x;
}
```

Pokemon → Class

↓

pikachu → objects of class pokemon

charizard → objects of class pokemon

mewtwo → objects of class pokemon

Ques : Create a structure type ‘Person’ with name, salary and age as its attributes. Declare and initialize 2 variables for this. Print the name of first person and age of the other.

Homework

```
struct Person {
    char name[50];
    int salary;
    int age;
};
```

How are structure elements stored?

↓

elements are stored in a continuous memory location

Typedef

&& the multiple pointer declaration problem

```
typedef old name newname;
```

Array of Structures

Why?

int arr[ ] → array of integers

char arr[ ] → array of characters / Strings

```
typedef struct pokemon {
        int hp;
        int attack;
        int speed;
        char tier;
    } pokemon;
```

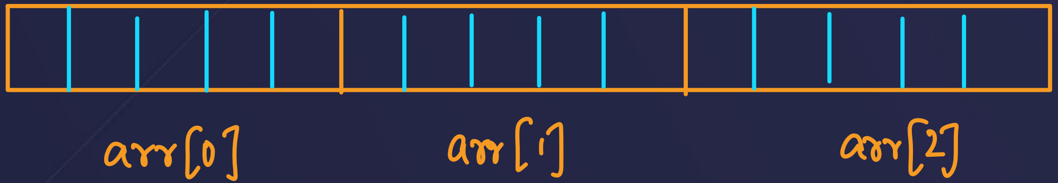
→ pokemon arr[ ] → array of pokemon

Array of Structures

Declaration and Access → arr[i] . attribute ;

```
typedef struct pokemon {
        int hp;
        int attack;
        int speed;
        char tier;
        char name[15];
} pokemon;

pokemon arr[3] ;
```



Ques : A record contains name of cricketer, his age, number of test matches that he has played and the average runs that he has scored in each test match. Create an array of structure to hold records of 20 such cricketer and then write a program to read these records

```
                                        Cricketer arr[20];

typedef struct cricketer{
    char name [20];

    int age;

    int noOfMatches;

    float average;
} cricketer;
```

Ques : State true or false

In an array of structures, not only are all structures stored in contiguous memory locations, but the elements of individual structures are also stored in contiguous locations.

→ True

Features of structures

Assigning the value of one structure variable to another of the same type

Ques : Create a structure 'date' that contains three members namely date, month and year. Create 2 structure variables with different dates and now compare the two. If the dates are equal then display message as "Equal" otherwise "Unequal".

```
typedef struct date {
        int date;
        int month;
        int year;
} date;
```

Ques : Now create another structure variable by assigning the first date to it. Compare the first and third dates.

followup

Features of structures

Nesting one structure within another structure

Ques :

maruti.engine.bolts = 25 ;

Which of the following is True?

1. structure bolts is nested within structure engine True
2. structure engine is nested within structure maruti True
3. structure maruti is nested within structure engine
4. structure maruti is nested within structure bolts

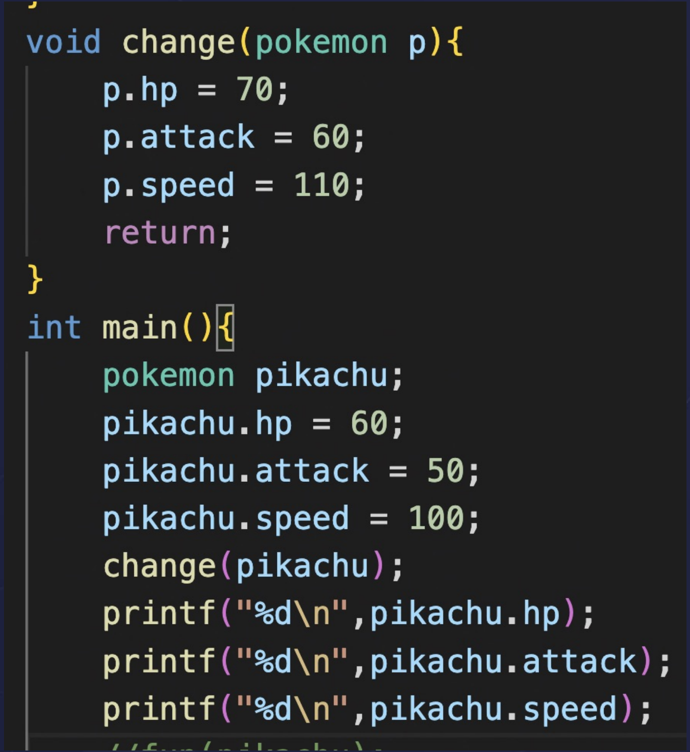
```
struct bolts {          struct engine {          struct maruti {

                               bolts b;                 engine e;
    }                        }                       {
```

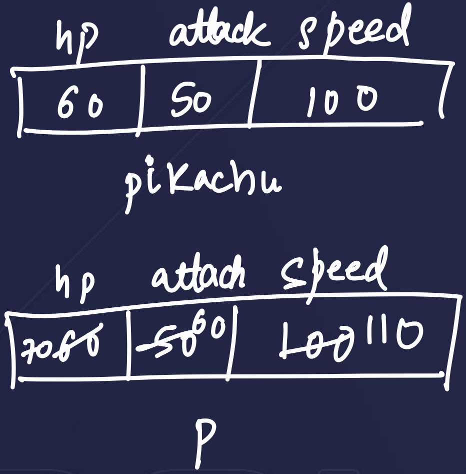
Features of structures

Important :

A structure variable can be passed to a function



```
}
void change(pokemon p){
    p.hp = 70;
    p.attack = 60;
    p.speed = 110;
    return;
}
int main(){
    pokemon pikachu;
    pikachu.hp = 60;
    pikachu.attack = 50;
    pikachu.speed = 100;
    change(pikachu);
    printf("%d\n",pikachu.hp);
    printf("%d\n",pikachu.attack);
    printf("%d\n",pikachu.speed);
    //fun(pikachu);
```



Structures are passed by value :

Ques : Create a structure to specify data on students with these attributes: Roll number, Name, Department, Course, Year of joining. Create 2 structure variables. Now, create a function to check if two students have the same Department. Pass the two structure variable as input to this function.

Homework

```
Struct student {
    int rno;
    char name[20];
    char dept [20];
    char course [30];
    int yearOf Joining;
};
```

```
void check (student s1, stud)
{
if (s1.dept == s2 dept):
```

```
Struct student s1, s2;

check (s1, s2);
```

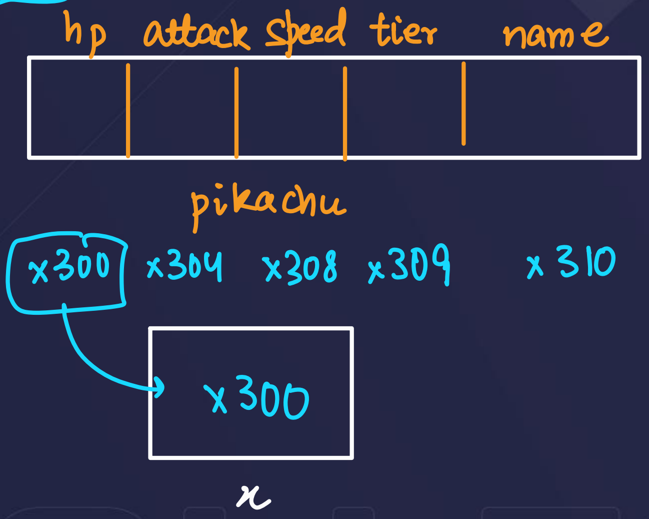
Features of structures

Structure pointers

Important

```
pokemon pikachu;

pokemon* x = &pikachu;
```



Ques : Create a structure ‘person’ having attributes as age and weight. Access its structure variables using pointers.

```
typedef struct person {
        int age;
        float weight;
} person;

person p1;

// p1.age = 23;
// p1.weight = 68.7;
```

```
person* x = &p1;

# Way of accessing/initialization/modifying
    (*x).age / x -> age
    (*x).weight / x -> weight
```

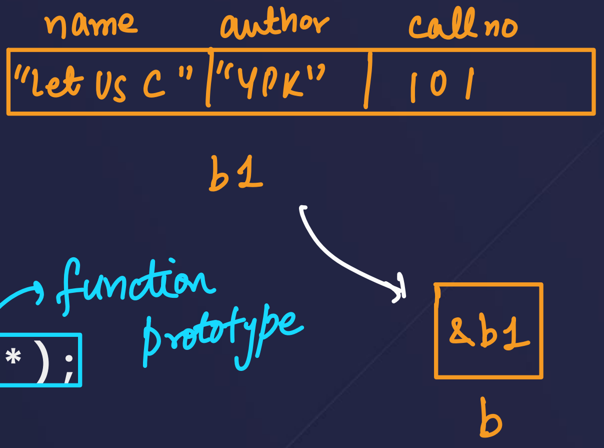
Ques : Predict the output:

```
# include <stdio.h>
struct book {
        char name[ 25 ] ;
        char author[ 25 ] ;
        int callno ;
} ;
void display ( struct book * ) ;

int main( ) {
        struct book b1 = { "Let us C", "YPK", 101 } ;
        display ( &b1 ) ;
        return 0 ;
}

void display ( struct book *b ) {
        printf ( "%s %s %d \n", b->name, b->author, b->callno ) ;
}
```

→ function prototype



b->name ↓ (*b).name

b->author ↓ (*b).author

b->callno ↓ (*b).callno

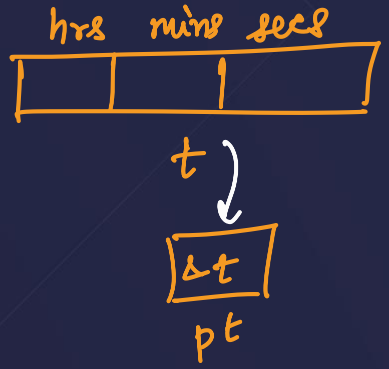
Output

Let us C YPK 101

Ques :

```
struct time {
      int hours ;
      int minutes ;
      int seconds ;
} t ;
struct time *pt ;
pt = &t ;
```

→ struct time t;



With reference to the above declarations which of the following refers to t's seconds correctly:

1. pt.seconds
2. (*pt).seconds ✓
3. time.seconds
4. pt -> seconds ✓

t . seconds ✓

(*pt) . seconds ✓

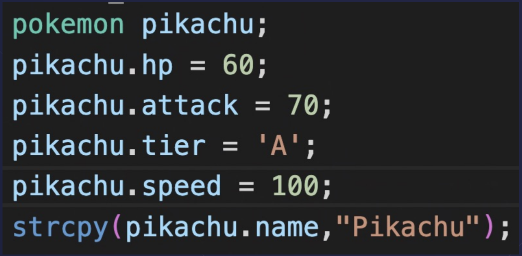
pt -> seconds ✓

Structures v/s Unions

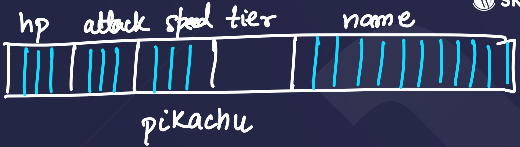
```
struct pokemon{
      int hp;
      int speed;
      int attack;
      char tier;
      char name[15];
};
```

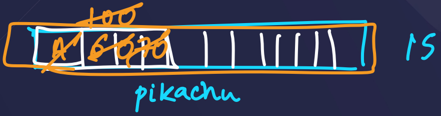
```
union pokemon{
      int hp;
      int speed;
      int attack;
      char tier;
      char name[15];
};
```

only one member can be used at a time.



```
pokemon pikachu;
pikachu.hp = 60;
pikachu.attack = 70;
pikachu.tier = 'A';
pikachu.speed = 100;
strcpy(pikachu.name,"Pikachu");
```





```
union
  ⇕
struct pokemon{
      int hp;       → 4 bytes
      int attack;   → 4 bytes
      int speed;    → 4 bytes
      char tier;    → 1 byte
      char name[15]; → 15 bytes
};
```

struct → 28 bytes

union → 15 bytes ✓