Review Question:

What is the name of the pointer shown below?

```
int *pointerToInt;
```

i. *pointerToInt
ii. pointerToInt
iii. int *pointerToInt
iv. pointertoint

Review Question:

How do you declare a pointer to

```
int i = 10;
```

?

i. int p
ii. int *p
iii. p
iv. *p
v. int &p
vi. &p

Review Question:

Which expression do you use to access the address of the variable weight:

```
float weight = 51.8;
```

?

i. *weight
ii. &weight
iii. weight

Review Question:

Assuming int i = 2, j = 1, k = 4, *p, *q, *r;

Is the following assignment legal (type correct):

```
p = i;
```

?

i. yes
ii. no

Review Question:

Assuming int i = 2, j = 1, k = 4, *p, *q, *r;

Is the following assignment legal (type correct):

```
*p = &i;
```

?

i. yes
ii. no

Review Question:

Assuming int i = 2, j = 1, k = 4, *p, *q, *r;

Is the following assignment legal (type correct):

```
p = *&q;
```

?

i. yes
ii. no

Review Question:

- What is the difference between the NULL pointer and the null character?

Short Answers (1)

1. (ii) pointerToInt
2. (ii) int *p, The emphasis is on the pointer declaration: with initialisation it would be int *p = &i;
3. (ii) &weight
4. (ii) no, p is a pointer to an int (holds a memory address) and i is an int (holds an int value)
5. (ii) no, *p is an int (the value kept at the address p points to) and &i is the address of i
6. (i) yes, p is a pointer (address), *&q is the value at address of q, which is also a pointer

Short Answers (2)

7. Although both terms use the word NULL the two are different: the NULL pointer ((void *)0) is used indicate a memory address that doesn't have a typical value, i.e. we say that such a pointer doesn't point 'anywhere'. If p is a NULL pointer, *if (!p)* returns true.

   The null character is '\0', a special character that is used to indicate the end of a string and is used by many string functions, such as strcpy, to work out the end of a string.

   (And 0 is an integral literal, 0.0 a floating point literal and '0' a character literal).

   See Tan et al pg. 345

Structures

- A structure is a collection of one or more variables,
- Possibly of different types,
- Grouped together under a single name
- So that they can be manipulated together.

- Think of examples:
  - A person
  - A student
  - The water consumption in a city

Accessing Members (1)

- Syntax:

  structure-name.member

PERSON Li={**19,50.6,'M',"Li"**};

```
printf("PERSON Li: the name is %s, the age is %d\n",
  Li.szName, Li.nAge);
```

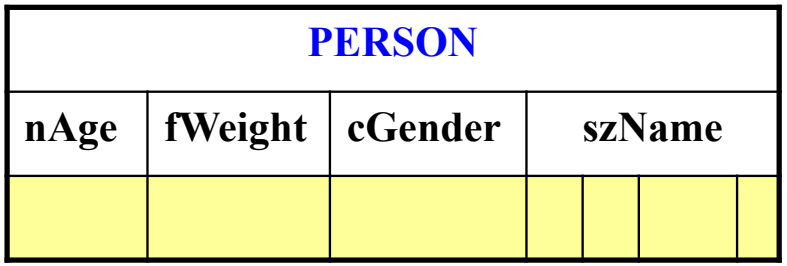
How would we refer to Li's:

i. gender and

ii. weight?

Practice 1

1. **Create a structure of type Person**



2. **Declare three variables of struct Person in different ways and initialize them.**

3. **Print their members.**

Pointers to Structures

- If pp points to a PERSON structure ...
- then (*pp).nAge and (*pp).cGender etc. are its members

Note:

- The parentheses are necessary in (*pp) because the precedence of . is higher than *
- *pp.nAge means *(pp. nAge), a pointer to the member nAge

Accessing Member Variables

Shorthand:

If pp is a pointer to a struct, the member variables can be accessed using ->

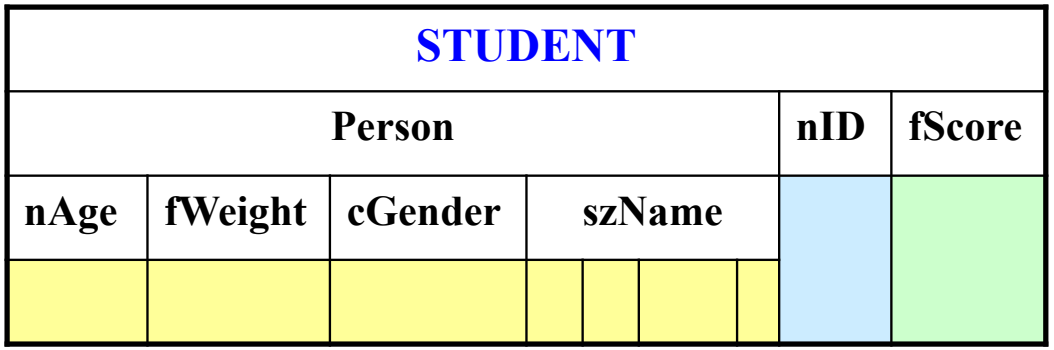
Example:

```
void printPerson(struct PERSON *pp) {
  printf("%d, %c)", pp->nAge, pp->cGender);
}
```

How could we access the members of a more complex structure?

Practice 2

1. **Create a structure of type STUDENT**

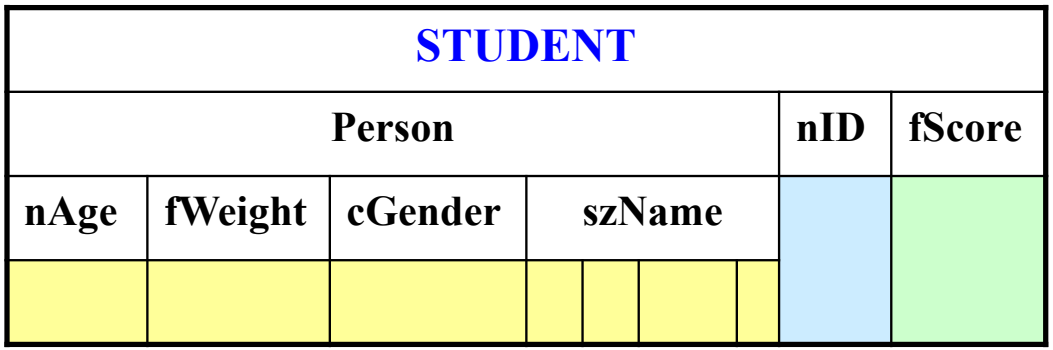


2. **Declare three variables of struct STUDENT and initialize them.**

3. **Print their members.**

Practice 3

1. **Write a function printStudent with a parameter of type pointer to struct STUDENT.**



2. **Use this function to print the details of one of the three struct STUDENT variables.**