POINTER ARITHMETIC

1. Pointers can be incremented or decremented to point to different locations like:

```
ptr1 = ptr2 + 3;
   ptr ++;
   -- ptr;
```

However, ptr++ will cause the pointer ptr to point the next address value of its type.

For example,

if ptr is a pointer to float with an initial value of 65526, then after the operation ptr ++ or ptr = ptr+1, the value of ptr would be 65530. Therefore, if we increment or decrement a pointer, its value is increased or decreased by the length of the data type that it points to.

2. If ptr1 and ptr2 are properly declared and initialized pointers, the following operations are valid:

```
res = res + *ptr1;
*ptr1 = *ptr2 + 5;
prod = *ptr1 * *ptr2;
quo = *ptr1 /  *ptr2;
```

Note that there is a blank space between / and * in the last statement because if you write /* together, then it will be considered as the beginning of a comment and the statement will fail.

3. Expressions like ptr1 == ptr2, ptr1 < ptr2, and ptr2 != ptr1 are permissible provided the pointers ptr1 and ptr2 refer to same and related variables. These comparisons are common in handling arrays. Suppose p1 and p2 are pointers to related variables. The following operations cannot work with respect to pointers:

1. Pointer variables cannot be added. For example, p1 = p1 + p2 is not valid.

2. Multiplication or division of a pointer with a constant is not allowed. For example, p1 * p2 or p2 / 5 are invalid.

3. An invalid pointer reference occurs when a pointer's value is referenced even though the pointer doesn't point to a valid block. Suppose p and q are two pointers. If we say, p = q; when q is uninitialized. The pointer p will then become uninitialized as well, and any reference to *p is an invalid pointer reference.