```c
#include <stdio.h>

int main(void)
{
    //variable declarations
    int num;
    int *ptr = NULL;
    int ans;

    //code
    num = 5;
    ptr = &num;

    printf("\n\n");
    printf(" num     = %d\n", num);
    printf(" &num    = %p\n", &num);
    printf(" *(&num) = %d\n", *(&num));
    printf(" ptr     = %p\n", ptr);
    printf(" *ptr    = %d\n", *ptr);

    printf("\n\n");


    // Add 10 to 'ptr' which is the address of 'num' ...
    // Hence, 10 will be added to the address of 'num' and the resultant address will be displayed
    printf("Answer Of (ptr + 10)  = %p\n", (ptr + 10));


    // Add 10 to 'ptr' which is the address of 'num' and give value at the new address...
    // Hence, 10 will be added to the address of 'num' and the value at resultant address will be displayed ...
    printf("Answer Of *(ptr + 10) = %d\n", *(ptr + 10));


    // Add 10 to '*ptr' which is the value at address of 'num' (i.e : 'num' i.e: 5) and give new value. without any change in any address ...
    // Hence, 10 will be added to the '*ptr' (num = 5) and the resultant value will be given (*ptr + 10) = (num + 10) = (5 + 10) = 15 ...
    printf("Answer Of (*ptr + 10) = %d\n\n", (*ptr + 10));


    // *** ASSOCIATIVITY OF * (VALUE AT ADDRESS) AND ++ AND -- OPERATORS IS FROM RIGHT TO LEFT ***


    // (RIGHT TO LEFT) Consider value *ptr ... Pre-increment *ptr ... That is, value at address 'ptr' i.e: *ptr is pre-incremented (++*ptr)
    ++*ptr; // *ptr is pre-incremented ... *ptr is 5 ... after execution of this statement ... *ptr = 6
    printf("Answer Of ++*ptr : %d\n", *ptr); //Brackets not necessary fo pre-
```

```
        increment / pre-decrement
45
46
47      // (RIGHT TO LEFT) Post-increment address ptr ... That is, address 'ptr' i.e:
          ptr is post-incremented (ptr++) and then the value at the new address is
          displayed (*ptr++)...
48      *ptr++; // Incorrect method of post-incrementing a value using pointer ...
49      printf("Answer Of *ptr++ : %d\n", *ptr); //Brackets ARE necessary fo post-
          increment / post-decrement
50
51
52      // (RIGHT TO LEFT) Post-increment value *ptr ... That is, value at address
          'ptr' i.e: *ptr is post-incremented (*ptr)++
53      ptr = &num;
54      (*ptr)++; // Correct method of post-incrementing a value using pointer ...
          *ptr is 6 ... at this statement *ptr remains 6 but at next statement *ptr =
          7 (post-increment)
55      printf("Answer Of (*ptr)++ : %d\n\n", *ptr); //Brackets are necessary fo post-
          increment / post-decrement
56
57      return(0);
58  }
59
60
```