Problem 6:-

```c
int a = 7;
int *aPtr = &a; // set aPtr to the address of a

printf("Address of a is %p\nValue of aPtr is %p\n\n", &a, aPtr);
printf("Value of a is %d\nValue of *aPtr is %d\n\n", a, *aPtr);
printf("Showing that * and & are complements of each other\n");
printf("&*aPtr = %p\n*&aPtr = %p\n", &*aPtr, *&aPtr);
```

Code Breakdown:

int a = 7;

- Declares an integer variable a and initializes it to 7.

---

int *aPtr = &a;

- Declares a pointer aPtr that stores the **address of a**.
- aPtr now points to the memory location where a is stored.

---

printf("Address of a is %p\nValue of aPtr is %p\n\n", &a, aPtr);

- This prints:
    - The address of a (using %p to format a pointer).
    - The value of aPtr, which should be the **same address** as &a.

Expected output :

Address of a is 0x7ffee1c7b68c

Value of aPtr is 0x7ffee1c7b68c

---

printf("Value of a is %d\nValue of *aPtr is %d\n\n", a, *aPtr);

- a is 7
- *aPtr dereferences the pointer — i.e., it gives the **value at the address** stored in aPtr, which is also 7.

Expected output:

Value of a is 7

Value of *aPtr is 7

---

printf("Showing that * and & are complements of each other \n");

- Just a message to introduce the next concept: * (dereferencing) and & (address-of) cancel each other out.

---

printf("&#aPtr = %p\n* &aPtr = %p\n", &aPtr, *&aPtr);

- &aPtr: the **address of the pointer aPtr** (i.e., where the pointer itself is stored in memory).
- *&aPtr: you take the address of aPtr, then dereference it — which brings you back to the **value of aPtr**.
- &aPtr: a different address from &a (because aPtr is a different variable).

- *&aPtr: this cancels out to just aPtr, which holds the **address of a**.

Expected output :

Showing that * and & are complements of each other

& aPtr = 0x7ffee1c7b680

* &aPtr = 0x7ffee1c7b68c

Why did we get that output?

Because:

- aPtr holds the address of a.
- *aPtr fetches the value at that address.
- &aPtr gives the location where the pointer is stored.
- *&aPtr cancels out to the original value of aPtr.