Problem 6 Explanation

Code:

```
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);
```

Output Explanation:

1. int a = 7;
   - Declares an integer variable a and initializes it with the value 7.

2. int *aPtr = &a;
   - Declares a pointer aPtr that stores the address of variable a.

3. printf("Address of a is %%p\nValue of aPtr is %%p\n\n", &a, aPtr);
   - Both &a and aPtr will print the same address because aPtr points to a.

4. printf("Value of a is %%d\nValue of *aPtr is %%d\n\n", a, *aPtr);
   - a holds the value 7.
   - *aPtr dereferences the pointer and gives the value stored at the address aPtr is pointing to, which is als

5. printf("&*aPtr = %%p\n*&aPtr = %%p\n", &*aPtr, *&aPtr);
   - &*aPtr means: dereference aPtr to get a, then take the address of a, which gives the original address. S

- *&aPtr means: take the address of aPtr, then dereference it. You get back the value of aPtr, which is ag

Conclusion:

This code demonstrates the relationship between pointers and variables, and how the * (dereference) and

* undoes &, and & undoes *, as shown in the final line.