1) Assume that the value of an int variable i is not zero. Do you think it is possible that \( i += 3*i \) will set the value of i to zero? If so, under what circumstances?

2) Consider an int variable x and consider a pointer float* p=(float*)&x pointing to x. If we store a float value 2.35 in x directly using \( x=2.35 \) or indirectly using \( *p=2.35 \), will we get the same bit pattern stored in x in both cases, or will we get different patterns?

3) In a program we allocated 20 bytes of memory using \( p = malloc(20) \), yet we only need 10. For the efficiency reason we decided to "return" a portion of the segment to the memory manager using \( free(&p[10]) \). What will happen? Will it compile? If so, will the program execute correctly and cause memory segment fragmentation? Or will it not work as intended --- the memory manager will not "accept" the memory and the program will continue executing? Or the program will be terminated?

4) In the following program we allocate a memory segment of 20 bytes. What will happen to the memory segment after the program terminates?

   ```c
   #include <stdlib.h>

   int main()
   {
       char *p;
       p = malloc(20);
       strcpy(p,"hello");
       return 0;
   }
   ```

5) What are the differences between the system stack and the system heap? Try to list all differences --- in their roles and purpose as well as in their implementations.

6) Let x, y be two integer variables, and let int* p. Let \( x = p[0] \) while \( y = *p \). Are the values of x and y the same? Under what circumstances they may differ?

7) Let int x[4] be an integer array. Let int* p, *q be pointers. Let \( p = x \) and \( q = &x[0] \). Are both pointers pointing to the same location or to different ones?