Question 1. (2pts) When you use `calloc()`, how is memory allocated? Circle the best answer.

(a) Memory is allocated on the runtime stack.
(b) Memory is allocated on the runtime heap.
(c) Both (a) and (b), depending on the arguments to `calloc()`.
(d) The operating system uses Snapchat, just to make sure the allocated memory doesn’t stick around too long.

Question 2. (2pts) Given dynamically allocated memory segment $M$ of size $n$ bytes, when you use `realloc()` to reallocate $M$ to be $q$ bytes, with $n \neq q$, what happens? Circle the best answer.

(a) Memory segment $M$ is either increased or decreased in size.
(b) Memory segment $M$ is discarded (i.e., `free()` is called) and new memory is dynamically allocated for $M$.
(c) The first $z$ bytes of memory segment $M$ are unchanged, where $z$ is the minimum of $n$ and $q$.
(d) All of the above are possible.
(e) None of the above are possible.
Question 3. (6pts) What is the exact terminal output of the code below? Assume a 64-bit architecture and that all system calls return successfully. Note that there are not compilation warnings or errors.

```
#include <stdio.h>
#include <stdlib.h>

#define Q 10

int main()
{
    int j;
    int * q = malloc( sizeof( int ) );
    *q = 1973;
    printf( "%d", (int)sizeof( int * ) );
    fprintf( stderr, "*q is %d!\n", *q );

    char * t = calloc( Q, sizeof( char ) );
    for ( j = 0 ; j < Q ; j++ )
        t[j] = 'Q';
    t[4] = '\0';

    printf( "%s\n", t );
    return EXIT_SUCCESS;
}
```