Computer Science 50: Introduction to Computer Science I
Harvard College
Fall 2007

Quiz 1

Solutions

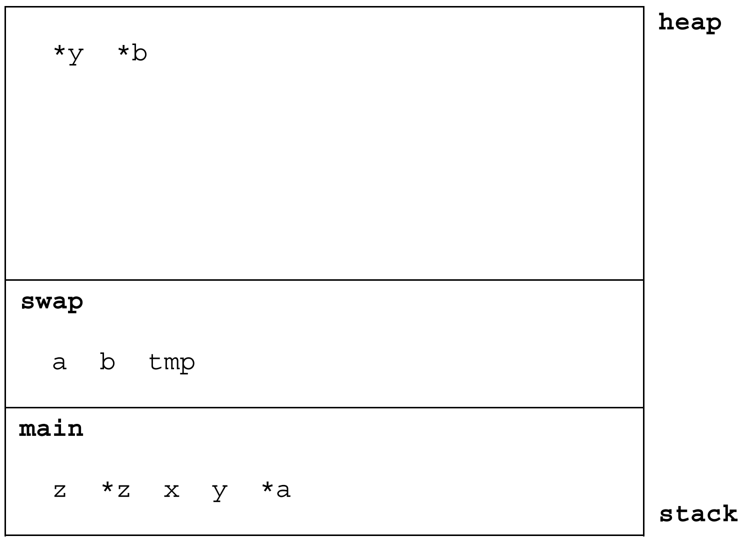
Answers other than the below may be possible.

Multiple Choice.

0. a, b, c, or d
1. d
2. c
3. b

Let's see how good your memory is.

4.



O(mega).

5. Even if its input is already sorted, this implementation iterates over *n* elements *n* times, for a total of $n \times n = n^2$ steps. If we instead check, at the end of each iteration of the outer loop, whether or not we actually swapped any neighbors, breaking out of that loop if not, we can keep our running time in $\Omega(n)$.

Should have memoized this!

6. 3

7. 9

This function is so foobar!

8. Factorial.

   For sufficiently large non-negative *n*, the function suffers from overflow, as *n*! exceeds `INT_MAX`.

   For sufficiently large non-negative *n*, `foobar` is recursively called so many times that the calls' frames overrun the available stack space.

```
int
foobar(int n)
{
    int answer = 1;
    for (int i = n; i > 1; i--)
        answer *= i;
    return answer;
}
```

Fun with Tables.

9.

	O	**Ω**	**assumptions, if any**
Binary Search	log *n*	1	input is a sorted array
Linear Search	*n*	1	
Merge Sort	*n* log *n*	*n* log *n*	
Selection Sort	n^2	n^2	

10.

base-2	**base-10**	**base-16**
1010	10	A
10	2	2
10000	16	10
10000000	128	80
111100000000	3840	F00
11111111	255	FF

11.

	`sizeof`
`char`	1
`char *`	4
`int`	4
`int *`	4
`long long`	8
`long long *`	4

Rapid Fire.

12. Sometimes recursion lends itself to more literal translation (and thus easier implementation) of algorithms into code (*e.g.*, divide-and-conquer strategies).

13. The algorithm is in both $\Omega(n)$ and $O(n)$.

14. If `foo` is a member of some `struct` called `bar` to which `baz` is a pointer, the dot operator allows you to access `foo` by way of `bar.foo`, whereas the `->` operator allows you to access the same by way of `baz->foo`.

15. As the number of cores in our computers increases but the number of things we, as humans, can do simultaneously remains roughly constant, we need software to take greater advantage of parallelism if we are to benefit, in terms of our computers' performance, from so many more cores.

16. "Deleting" a file often means "forgetting" where its bits are (by modifying the file's directory entry) but not actually overwriting them. By searching a hard drive for modified directory entries or known signatures, files can often be recovered.

17. A memory leak is the result of some program requesting memory of the operating system (as via `malloc`) and never freeing it (as via `free`), even when no longer needed.

Makin' Copies.

18. Because `s2` equals `s1`, it, as a pointer to a `char`, merely points to the same `char` in memory as does `s1`. Accordingly, `s1[0]` and `s2[0]` represent the same `char` in memory. Capitalizing one thus capitalizes the other.

19.

```
string
CopyString(string s)
{
    if (s == NULL)
        return NULL;
    string t = (string) malloc((strlen(s) + 1) * sizeof(char));
    if (t == NULL)
        return NULL;
    for (int i = 0, n = strlen(s); i <= n; i++)
        t[i] = s[i];
    return t;
}
```

LAST ONE!

20.

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

int
main(int argc, char * argv[])
{
    // try to open input
    FILE * fin = fopen(argv[1], "r");
    if (fin == NULL)
        return 1;

    // prepare output's name
    for (int i = 0, n = strlen(argv[1]); i < n; i++)
        argv[1][i] = toupper(argv[1][i]);

    // try to open output
    FILE * fout = fopen(argv[1], "w");
    if (fout == NULL)
    {
        fclose(fin);
        return 1;
    }

    // convert input's contents to uppercase
    char c;
    while (!feof(fin))
    {
        fread(&c, sizeof(char), 1, fin);
        c = toupper(c);
        fwrite(&c, sizeof(char), 1, fout);
    }

    // that's all folks
    fclose(fout);
    fclose(fin);
}
```