Worksheet 2

Overview

The w orkshop for Week 3 w ill start w ith a mini-tutorial on computational complexity and static/dynamic arrays.

Tutorial Questions

Question 2.1 Given the follow ing functions *f(n)* and *g(n)*, is *f* in *O(g(n))* or is *f* in *Θ(g(n))*, or both?

	f(n)	**g(n)**	**f in O(g(n))**	**f in Θ(g(n))**	**both**
(a)	n + 100	n + 200			
(b)	log2(n)	log10(n)			
(c)	2^n	2^(n + 1)			
(d)	2^n	3^n			
Question 2.2 big-O notation gives an upper bound for a function. In Computer Science, w e often loosely use big-O to mean the *least* upper bound.					

First of all: make sure you can describe the difference betw een *any* upper bound and the *least* upper bound. big-Θ (theta) is an exact bound: the function is bounded form below and from above by g(n).

The follow ing table uses big-O notation and big-Θ notation to approximate the running time of algorithms. Given the descriptions of the run time in the tw o left hand columns, w hat can you say, in each instance, about the relative performance of the tw o algorithms w hen:

1. big-O is used in the usual Computer Science sense, as the least upper bound;
2. big-O is used in its strictest sense to mean any upper bound.

Algorithm 1	**Algorithm 2**	**Relative Performance: CS sense of big-O**	**Relative Performance: Strict sense of big-O**	
O(n log n)	O(n^3)			
Θ(n log n)	O(n^3)			
O(n log n)	Θ(n^3)			
Θ(n log n)	Θ(n^3)			
Question 2.3 What is the difference betw een the follow ing declarations? (What do they each declare)				

`int a[10][20];`				

```
int *b[10];
```

1. How could you use them both as 2-dimensional arrays? (Show what you could do by modifying the code below so a and b print the same values)

NOTE: You'll also need to clean up at the end to ensure no false alarms in code checking.

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

int main(int argc, char **argv){
    int i,j;
    int a[10][20];
    int *b[10];

    /* Write code here to use b in the below loops. */

    for(i = 0; i < 10; i++){
        for(j = 0; j < 20; j++){
            /* This fills a and b with the numbers 0 -> 199 */
            a[i][j] = i*20 + j;
            b[i][j] = i*20 + j;
        }
    }

    for(i = 0; i < 10; i++){
        for(j = 0; j < 20; j++){
            /* Feel free to modify this to do different things, only prints
            when i == j to give less output, no real significance. */
            if(i == j){
                printf("a[%d][%d] = %d, b[%d][%d] = %d\n",i,j,a[i][j],i,j,b[i][j]);
            }
        }
    }

    return 0;
}
```

1. What advantages might there be to declaring an array like *b[] above, instead of like a[][] above?

2. How could you make the variable declared as int ** into a 2-dimensional array? (Write the code below.)

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

int main(int argc, char **argv){
    int i,j;
    int a[10][20];
    int *b[10];
    int **c;

    /* Take your code from part 1 for b and put it here. */

    /* Write the code for initialising c and put it here. */

    for(i = 0; i < 10; i++){
        for(j = 0; j < 20; j++){
            /* This fills a and b with the numbers 0 -> 199 */
            a[i][j] = i*20 + j;
            b[i][j] = i*20 + j;
            c[i][j] = i*20 + j;
        }
    }

    for(i = 0; i < 10; i++){
        for(j = 0; j < 20; j++){
            /* Feel free to modify this to do different things, only prints
            when i == j to give less output, no real significance. */
            if(i == j){
                printf("a[%d][%d] = %d, b[%d][%d] = %d, c[%d][%d] = %d\n",i,j,a[i][j],i,j,b[i][j],i,j,c[i][j]);
            }
        }
    }

    return 0;
}
```

Question 2.4 Give *a* characterisation, in terms of big-O, big-Ω and big-Θ of the following loops:

```
int p = 1, i;
for(i = 0; i < 2*n; i++){
    p = p * i;
}
```

Big-O:
Big-Ω:
Big-Θ:

```
int s = 0, i, j;
for(i = 0; i < 2*n; i++){
    for(j = 0; j < i; j++){
        s = s + i;
    }
}
```

Big-O:
Big-Ω:
Big-Θ:

Programming Exercises

Programming 2.1 In this task, the objective is simply to write a program which changes the value of a variable in a different scope.

This problem is intended to highlight some common pointer misconceptions and ensure you have a strong grasp on how things are actually working. The task is fairly simple if you already have a very strong grasp of pointers, but might raise some eyebrows if you think the C programming language is more complex than it actually is or don't quite fully understand how c handles scoping.

Note: In this exercise, you can make changes ***anywhere*** *to make this program work, the intention is just to fix this program to do what it is supposed to do. So changeX after is reflected in main after.*

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

/* You may have to make some changes to this program to make this work. */
/* Changes the value of x given to the integer 9. */
void changeX(int x);

int main(int argc, char **argv){
    int x = 5;
    printf("[main    before] x = %d\n",x);
    changeX(x);
    printf("[main    after ] x = %d\n",x);
    return 0;
}

void changeX(int x){
    printf("[changeX before] x = %d\n",x);
    x = 9;
    printf("[changeX after ] x = %d\n",x);
}
```

Programming 2.2 In this task, the objective is to write a dynamically allocated array that takes a number of words and stores these in an array.

Input for this program will be first a number explaining how many strings will be written to stdin.
Then, for each of these strings, a number representing the length of the string will be written to stdin, followed by some whitespace (ignored) and the string to be stored.

This array should be declared dynamically using malloc. A bit of scaffolding has been provided to constrain the problem a little.

Note: If you didn't complete Programming 1.2 from Week 2's Worksheet 1, you might find that a useful place to start.

```
//%stdin: "5 4 this 2 is 4 some 7 example 4 text"
//%stdout: "this is some example text"
//%memtotalnoterm: 21
//%memtotal: 26
//%memaux: 40
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

int main(int argc, char **argv){
    int numStrings;
    int nextStringLength;
    int i, j;
    char **strings;

    /* Read number of strings to store. */

    /* Allocate space for the array of strings. */

    /* For each string, get its length, allocate space for it
        and read all the characters into the string.
        Note: Remember, these are standard strings. */


    /* The below prints all words in the array with spaces between them. */
    if(numStrings > 0){
        printf("%s",strings[0]);
    }

    for(i = 1; i < numStrings; i++){
        printf(" %s",strings[i]);
    }
    printf("\n");

    return 0;
}
```

Programming Challenge 2.1 In this challenge, you work at a company called Hidebound Inc. and you've been hired as an unpaid intern to w ork on one of the corporation's most important products. As upper management are overly w orried about any issues occuring as a result of an unpaid intern's w ork, they have tasked you w ith only a very small part of the program.

The task for the part of the program you are required to w rite is the function modifyImportant, w hich modifies the value of the variable in the main function called important.

You may do w hatever you like to achieve this end, some of these w ill be more portable, but the company is more interested in results, so w hatever achieves this *without* modifying any part of the code except the body of the function (clearly marked by comments) w ill be sufficient. Less options are available to you in the notebook (to the point that I'm not sure this is problem is solvable w ithout being *very* lucky), so you may find the university servers make the problem easier to solve.

```
#include <stdio.h>
#include <stdlib.h>
/* You may also include additional libraries, but I don't think you will need to. */

void changeImportant(int newVal);

int main(int argc, char **argv){
    int important = 5;
    printf("Value of important[%p] before: %d\n", &important, important);
    changeImportant(7);
    printf("Value of important[%p] after: %d\n", &important, important);
    return 0;
}

void changeImportant(int newVal){
    /* ---- Your jurisdiction starts here. ---- */
    /*
        Change the value of important in main to the newVal.
    */
    /* ---- Your jurisdiction ends here. ---- */
}
```

Programming Challenge 2.2 The technology stack at Hidebound Inc. uses a subset of C w hich doesn't have the '.' or '->' operators, as the higher-ups heard shortcuts like this w ere useful in an activity called "code golfing" and, misunderstanding w hat that meant, w anted to discourage all recreational activities on company time. The change improved compile times and required resources slightly so the developer in charge of that performance w as happy to force the change on the other programmers in the company. In this challenge, you'll need to replace a piece of code w hich does this using both the simple '->' and '.' operators w ith a piece of code that instead changes the value in the struct by using value casting and pointer addition instead.

This challenge is intended to highlight that '.' and '->' are merely shortcuts to other dereference operations and though you w ill eventually find your code is less messy w hen using them, understanding exactly w hat you are doing w ill reduce the number of errors you make and allow you to examine code closely w hen you have something complicated that isn't doing exactly w hat you think it should be. You may find reading through the (2nd) extra w orkshop material document on the LMS under the Resources section is particularly useful for this task.

As a hint, you may find the offsetof macro useful (you can find this using the *man* pages). For an extra challenge, try only using the sizeof macro, the address of operator (&) and the dereference operator (*). Note also that for the latter, a process know n as "packing" may sometimes add holes to structs w hich are unused, though that has been carefully avoided in the struct defined here.

```
/*
    This program was written by Richard Chad Sparrow
    as a test case for AB-testing the hazard management
    system.
*/
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>

struct hazard {
    char *description;
```

```c
    void *extraData;
    int extraDataType;
    int id;
    char severityClass;
};

void printHazard(struct hazard *hazard);

int main(int argc, char **argv){
    struct hazard hazard1;
    struct hazard hazard2;
    struct hazard *lastHazard;

    /* Hazard data setup. */
    hazard1.description = "Brake service required.";
    hazard1.extraData = NULL;
    hazard1.extraDataType = 0;
    hazard1.id = 1;
    hazard1.severityClass = 'A';

    hazard2.description = "Unknown issue in fluid level.";
    hazard2.extraData = NULL;
    hazard2.extraDataType = 0;
    hazard2.id = 2;
    hazard2.severityClass = 'U';

    lastHazard = &hazard2;

    printf("Hazards after setup:\n");
    printHazard(&hazard1);
    printHazard(&hazard2);

    /*
        The brake service hazard has been present for multiple
        services, so needs to be updated to severity class 'B'.
    */
    /* Original: hazard1.severityClass = 'B'; */
    /* CHANGE THE CODE HERE: */
    hazard1.severityClass = 'B';

    printf("Hazard 1 after class B severity update:\n");
    printHazard(&hazard1);

    /*
        The next hazard to be evaluted has been evaluated and
        its severity class has been found to be quite serious,
        class 'D'. As part of this issue, the id has also been
        increased to 3 and the hazard description has been
        changed to "Fluid leak in tank 4".
    */
    /* Original: lastHazard->severityClass = 'D'; */
    /* CHANGE THE CODE HERE: */
    lastHazard->severityClass = 'D';

    /* Original: lastHazard->description = "Fluid leak in tank 4"; */
    /* CHANGE THE CODE HERE: */
    lastHazard->description = "Fluid leak in tank 4";

    printf("Hazard 2 after description and D-class update:\n");
    printHazard(&hazard2);

    return 0;
}

void printHazard(struct hazard *hazard){

    printf("Hazard %d: %s [Class %c, extraDataType: %d]\n",
        hazard->id, hazard->description, hazard->severityClass,
        hazard->extraDataType);
}
```