CSC 352 : Study Guide for Exam 2

Ultimately, the topics for the final exam includes anything from class, videos, the readings, the PAs, and any other material covered up-to July 20th. This list is just provided as a guide.

Topics covered:

- Topics list from the previous study guide.
- C programming
  - This includes everything covered about C in this course. Language basics, types, functions, I/O, pointers, l-values and r-values, etc
- Standard library functions
  - You don't have to know every standard library function in the whole library, but you should be familiar with ones used in class and homework assignments. For example, ones from **`stdio.h`** such as printf, fgets, scanf, etc.
- General UNIX file system knowledge
- GDB
- File permissions
- Dynamic memory allocation
- Stack and heap
- malloc, calloc, free, etc
- Managing memory
- Valgrind
- Data structures in C (linked lists, trees, graphs, dynamically-sized arrays, etc)
- Make
- System Calls, strace

Problems

Solutions for these problems are not included in this guide. In order to check if your solutions are correct you can try:

1. Analyzing your solutions yourself after writing
2. Comparing with other students
3. For coding problems: Run and test the code!

For some of the coding questions, you can try implementing two versions - one with the standard library functions allowed (which could make it shorter) and another version with no standard library. You should not rely on only this study guide for exam preparation.

Problem 1

In this problem, you should write a C function named **`bubble_sort`**. This function should have one parameter of type **`StringLLNode*`**. **`StringLLNode`** represents a node for a linked list of strings, which is defined as:

```
typedef struct StringLLNode {
  char* string;
  struct StringLLNode* next;
} StringLLNode;
```

You can expect that the node that gets passed into the function represents the root of a linked list. The function should perform a bubble-sort on the linked list to get the elements into alphabetical order, based on the string contents in each node. Initially, try implementing this *with the standard* library functions. For example, you could use the **`strcmp`** function for comparing if a string is greater than the other. After implementing this, try accomplishing it *without* any help from standard library functions for additional practice.

Problem 2

Pretend that we have a simple game program called statistics whose source code has the following structure and characteristics:

```
- statistics
   |- statistics.c
   |- coremath.c
   |- coremath.h
   |- calc.c
   |- calc.h
   +- makefile
```

Expect that statistics.c contains the main function. coremath.c contains a collection of function implementations whose prototypes are in coremath.h and calc.c contains a collection of function implementations whose prototypes are in calc.h. statistics.c only includes calc.h and a few standard library header files. calc.c includes only coremath.h. coremath.h does not contain any other includes. When built, the resulting game executable should be named statistics. Write what should go in the makefile that has one rule per output file produced.

Problem 3

Provided the struct below, what is the minimum number of bytes of heap memory that will be allocated by calling `GroceryItem* node = malloc(sizeof(GroceryItem));`?

```c
typedef struct GroceryItem {
      char item[50];
      int itemIdNumber;
      float price;
} GroceryItem;
```

Problem 4

When we run a program with valgrind, it can sometimes display this type of problem

```
definitely lost: X bytes in Y blocks
```

and it can also display this type of problem:

```
Conditional jump or move depends on uninitialised value(s)
```

What do each of these mean? What is the difference between the two? Give an example of each.

Problem 5

Recall that behind-the-scenes on a UNIX system, a file is represented by an i-Node. For this problem, assume the following:

- A data block is 160 bytes.
- A data block pointer is 8 bytes.
- An i-Node will generally contain some amount of meta-data at the beginning, such as file permissions, last modified time, etc.
- The i-Node will have 5 direct pointers to data blocks.
- The i-Node will have 1 pointer to a data block that contains pointers to data blocks. (1 level of indirection)
- The i-Node will have 1 pointer to a data block that contains pointers to other data blocks, with pointers to data blocks. (2 levels of indirection)

Say we want to save an ASCII text file that contains 3,000 letters. How many total data block pointers will need to be used to store this? Show your work.

Problem 6

There are at least two problems with the following piece of code. Identify the problems and provide a fix for each problem.

```
int main(void) {
    char *name;
    for (int i = 0; i < 10; i++) {
        name = calloc(1000, sizeof(char));
        printf("Enter your name: ");
        scanf("%s", &name);
        printf("Your name is: %s\n", name);
    }
    free(name);
}
```

Problem 7

In this problem, you should write a C function named **`build_binary_tree`**. This function should have one parameter of type **`char*`** that represents the name of a file to read from. You can expect that the file will have exactly one word per line, and no word will be longer than 25 characters. The function is responsible for constructing a binary tree from these words, and should return the root node of the constructed tree, which should be a **`StringTreeNode*`**. You can expect that a **`StringTreeNode`** is defined as:

```c
typedef struct StringTreeNode {
  char* string;
  struct StringTreeNode* left;
  struct StringTreeNode* right;
} StringTreeNode;
```

The function should read through the strings from the file in order, and add each to the tree. Words that come earlier in alphabetical order should go to the left, and words that go later in alphabetical order should go to the right.

For example, if the input file contained the words:

```
dead
zebra
yelp
britain
america
zoo
comic
```

The resulting tree should be:

```
             dead
            /     \
     britain       zebra
    /     \         /   \
america   comic  yelp  zoo
```

Problem 8

The following function is supposed to add a node to the head of the linked list pointed at by *hd*.

```
struct node {
  int val;
  struct node *next;
}
void addNode(struct node *hd, int val) {
  struct node *newNode = malloc(sizeof(struct node));
  newNode->val  = val;
  newNode->next = hd;
  hd = newNode;
}
```

assuming the following declarations:

```
int x;
struct node *hd = NULL;
```

the function is called like:
x = *some integer*
addNode(hd, x);

The programmer is surprised that after making several such calls, the linked list is empty. Explain what is happening and write a fix.

Problem 9

After running valgrind by typing:

```
valgrind executable test1.txt
```

it shows no memory errors or leaks. Does this mean my program definitely has no memory errors or leaks? Why or why not?

Problem 10

I am writing a program with multiple files: main.c, treeUtils.c, treeUtils.h, printUtils.c, printUtils.h. The main function is inside of main.c and the executable should be named main. main.c depends on functions and definitions in both treeUtils.h and printUtils.h. However treeUtils and printUtils do not depend on each other. i.e. printUtils does not call any functions or use any definitions in treeUtils and vice versa. Write a makefile to create the executable main. Your makefile should only recompile files when it is necessary. i.e. do not write a makefile that always recompiles all files no matter what. Use the best practices taught in class.

Problem 11

What is the output of the following program?

```
#include
int main()
{
  int i = 9;
  int *iptr = &i;
  char *cptr = (char *) iptr;
  ++cptr;
  ++iptr;
  if ((unsigned long) cptr == (unsigned long) iptr)
    printf("equal\n");
  else
    printf("not equal\n");
  return 0;
}
```

Problem 12

(a) What is a file ? Explain file handling in C language.
(b) What does #include do? Explain.
(c) Explain about fprint() and fscanf() functions with suitable examples.
(d) What is Array ? Explain declaration, initialization and accessing elements of one and two dimensions of numeric Array.
(e) Write a program to concatenate one string to another string without using the library function.

Problem 13

Write a function in C which is named `construct_linked_list` with will have one string `char*` parameter that can be the name of a text field. In this text field, it will contain one or more lines where each line has a single integer number (in ASCII). The function is supposed to read in the numbers of the field and construct a simple linked list which contains each of these numbers in the sequence that they appear in the field. Use the following struct:

```
typedef struct ListNode {
  int value;
  Struct ListNode* next;
}
```

Problem 14

Write a C program having an overload function name `area()` to calculate the area of shapes like triangle ,square, circle.

Problem 15

Write a C program which has a function named `CopyContents`. This function should have two parameter of type `char*` that represents the name of a file to read from and to be copied from. You can expect that the file from which the contents is being read from will have exactly one word per line, and no word will be longer than 100 characters. This function will copy the contents from the read file to the second file.

Problem 16

In this problem, you should write a C function named `merge_numbers`. This function should have two parameters of type `char*` which you can expect will both be paths to text files containing an equal number of integer numbers values 0 - one billion, one number per line. This function should create a file named `numbers.bin`. The function should interleave the numbers from the two text files into this binary file, and write each in its 32 bit binary representation.

Problem 17

What is the role of (break, run, next/step, bt, frame, print) key option for gdb? Provide a 1-2 sentence description of what each command does.

Problem 18

How many bytes of heap memory does the below program allocate in total? Show your work.

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

int main() {
  uint8_t num = 0;
  uint8_t num2 = 1;
  while (num < num2) {
    num = num+1;
    num2 = num2+1;
  }
  for (int i = 0; i < num; i++) {
    char * x = malloc(sizeof(int32_t));
  }
  while (num > 0) {
    num = num / 2;
    char * z = calloc(2, sizeof(int16_t));
  }
  return 0;
}
```

Problem 19

Describe the difference between a standard library function and a system call. What are the key differences? Do system calls run on the user stack? What tool can be used to check what system calls a process calls?

Problem 20

Use strace to run your strep program from the homework assignment. List the top 5 system calls that it uses, and provide a 1-2 sentence description of what each system call does. HINT: Use the man pages to figure out what each does! Run:

```
$ man 2 syscall_name
```