CS 300 Exam 3 Review

Topics

1. Dynamic Memory
2. Linked Lists
3. Generic Programming
4. Hash Tables
5. Handles
6. Complexity
7. Bit Manipulation
8. Makefiles
9. Text files, argc, argv

Given the following program

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #define MAX_CHARS 4
5  char *getStr (char **hStr, int *size)
6  {
7    char name[MAX_CHARS];
8
9    printf ("Enter name: ");
10   scanf ("%s", name);
11   *size = strlen (name);
12   *hStr = (char *) malloc (*size);
13   strcpy (*hStr, name);
14   return *hStr;
15 }
16 int main ()
17 {
18   char *pName;
19   int size;
20
21   pName = getStr (&pName, &size);
22   puts (pName);
23
24   return EXIT_SUCCESS;
25 }
```

1. Draw a picture of the runtime stack right before statement 14 is executed.
2. Are there any potential execution problems if the name Bill is entered from the keyboard? If so, explain.
3. In function getStr, what is hStr? *hStr? **hStr? size? *size?
4. Is Valgrind cool with everything if Bill is typed in at the keyboard? If not, explain.

5. Consider the following data structure

```
typedef struct ListElement *ListElementPtr;
typedef struct ListElement
{
  void *pData;
  ListElementPtr psNext;
} ListElement;
```

6. Write a function that will concatenate one list on to the end of the other list
   void lstConcatenate (ListElementPtr *hsLeft, ListElementPtr *hsRight);

   After the concatenation, hsRight is set to NULL

   ex: hsLeft: 1->2->9 hsRight: 3->7->19->100
   result: hsLeft: 1->2->9->3->7->19->100 hsRight-|

7. What's the complexity of your function in 6. Why?

8. Finish the in class labs and make sure you understand the code in each lab.

9. Explain what a collision is during hashing.

10. Describe how linear probing can lead to poor access times for a Hash Table.

11. Hash the keys M6, G7, Q21, Y77, R19, Z18, and F2 using the hash formula h(Kn) = n mod 7 with the following collision handling technique:

    1. linear probing
    2. chaining
    3. Compute the average number of probes to find an arbitrary key for each method

12. The tool gpref (http://www.gnu.org/software/gperf/) will generate C code that produces a perfect hash function. However, gpref requires that you provide every key you will ever hash. Why is this necessary?

13. Using the definition of big-O, show that the computing complexity of $N^2+\log_2 N+4$ is $O(N^2)$

14. An ordered list is pointed to by a pointer psList. The ordering is characters from smallest ASCII value to the largest ASCII value. Can we search this list for a specific character and return a pointer to the node, if the character exists, in $O(\log_2 N)$ time? Why or why not?

15. Using the List ADT, how would you implement the Stack functions?

16. A data file numbers.txt contains an unknown number of integers, one per line. Write a C program that opens the file and finds the average of all numbers read. The executable and data file both exist in the same directory. Further, the name of the data file is passed in through argv.

17. How does mid-square hashing differ from division hashing when determining the hash table size?