**CptS 122 – Data Structures**

**Summer 2021**

***Take-home Exam 1***

Friday, June 25, 2021

**Your Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ID Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**TA’s Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

READ THE FOLLOWING INSTRUCTIONS:

This exam is take-home. It should take ~ 1 hour to complete. You must work **individually** on this exam. You may use your book and notes if necessary. Please either handwrite or type your answers into this document. Late exam solutions will **not** be accepted! **Please show all work!!! Also, finding a solution online for any given problem could lead to zero credit for the problem and the exam.**

You must submit the exam through Blackboard by Monday, June 28, midnight PST. Be sure to answer each question precisely. Do not provide superfluous details in your answers. *NOTE: you do not need to comment your code solutions.*

|  |  |  |
| --- | --- | --- |
| **Part** | **Points Possible** | **Points Earned** |
| I | 30 |  |
| II | 70 |  |
| **Total** | **100** |  |

***Part I: Conceptual Questions (30 pts) – Short answer, Fill-in-the-blank, and Multiple-choice***

1. (4 pts) Compare and contrast dynamic linked queues and dynamic linked stacks; be thorough with your explanation. Also, provide example applications for each data structure.
2. (2 pts - 1 pt/blank) The function \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ supports the creation, maintenance, and destruction of a called function’s local variables, including parameters. It is placed in RAM and monitored closely by the system.
3. (2 pts) A region in memory that is not managed by the system, but by the programmer, is called the \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
4. (2 pts) A(n) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is a specialized version of a linked list in which nodes can be inserted only at the *start* of the list and deleted only from the *end* of the list.
5. (3 pts) (circle one) If you wanted to build an application that stores words in dictionary ordering, which data structure is most appropriate?
   1. Linked list b. Queue c. Stack d. Struct
6. (3 pts) (circle one) I have implemented a queue with a linked list, keeping track of a front node and a rear node with two pointer variables. Which of these pointer variables will change during an insertion into a NONEMPTY queue?
   1. Both change
   2. Only rear changes
   3. Only front changes
   4. Neither changes
7. (3 pts) (circle one) One difference between a queue and a stack is:
   1. Queues require linked lists, but stacks do not.
   2. Stacks require linked lists, but queues do not.
   3. Queues use two ends of the structure; stacks use only one.
   4. Stacks use two ends of the structure, queues use only one.
8. (3 pts) (circle one) If you wanted to build an application that performs “backtracking” in a maze so that the beginning of a path can be recovered if the wrong path is taken, which data structure is most appropriate?
   1. General linked list b. Queue c. Stack d. Binary search tree
9. (3 pts) (circle one) (circle one) What kind of list is best to answer questions such as "What is the item at position n?"

a. Lists implemented with an array

b. Singly-linked lists that are created dynamically

c. You cannot solve this problem with a list

1. (3 pts) (circle one) The following code fragment results in a \_\_\_\_\_\_\_\_\_\_\_\_\_\_.

a. memory leak b. dangling pointer c. memory freeze d. unresolved pointer

…..

pMem = (Node \*) malloc (sizeof (Node));

…..

pTemp = pMem;

free (pTemp);

…..

copyData = pMem -> data;

1. (2 pts) True/False (circle one): The following snippet of code will result in a runtime error.

Assume Node is defined as:

typedef struct node

{

char \* pData;

struct node \*pNext;

} Node;

…

Node \*pNode = NULL;

pNode = (Node \*) malloc (sizeof (Node));

strcpy (pNode -> pData, “Test item”);

***Part II: Programming Questions (70 pts) – Write C code***

1. (18 pts) Write a C function called compareLists() that compares the data (ASCII comparisons) in two singly linked lists of characters. Each list is a linked list version of a string, where each character is stored in a single Node. This function should operate similarly to strcmp(). However, a null character (‘\0’) does NOT need be stored in each list to represent the end. The function returns 0 if both lists (strings) are the same, 1 if the first list is greater than the second one, and -1 if the first list is less than the second one. Remember: all comparisons are based on ASCII values, not the number of nodes in the list. Note: the lists may be empty.

Assume that struct node is defined as follows:

typedef struct node

{

char data;

struct node \*pNext;

} Node;

Assume that struct list is defined as follows:

typedef struct list

{

Node \*pHead;

} List;

Use the following function header:

int compareLists(List list1, List list2)

{

1. (12 pts) Write a C function called concatenateLists() that concatenates/appends two linked lists. This function will append the source list to the end of the destination list. You do not need to maintain an order when the two lists are appended. Note: the lists may be empty.

Assume that struct node is defined as follows:

typedef struct node

{

double salary;

struct node \*pNext;

} Node;

1. (12 pts) Write a pseudocode algorithm that uses two stacks to simulate one queue. Only consider insertions and deletions from the queue. Since this is pseudocode, you do not have to describe how to implement push and pop (these are the operations that you will discuss to simulate the queue). However, you may assume that these are acceptable terms to use within your pseudocode algorithm. Make sure that the algorithm clearly describes what needs to be done with the stack during insertions and deletions from the queue.
2. (20 pts) Write a C function deleteNode() for a dynamic *doubly* linked list that has the following header:

int deleteNode (Node \*\*pList, char \*pSearchStr);

The function should delete the node with the string that matches the string pointed to by pSearchStr. If a match is found, then the function returns 1, otherwise the function returns 0.

Assume that struct node is defined as follows:

typedef struct node

{

char data[100]; // This will be used to store strings

struct node \*pNext;

struct node \*pPrev;

} Node;

1. (8 pts) Write a function called push(), which accepts a pointer to a pointer to a Node (representing the top of the stack) and the *character* to insert as parameters. The function should allocate memory for a Node, initialize the Node with the correct data, and insert it onto the top of the stack. The function needs to return 1 if memory was allocated successfully; 0 otherwise. You may NOT assume that a makeNode() function exists. The node is defined below:

typdef struct node

{

char data;

struct node \*pNext;

} Node;

Be sure to write out the function header!