COP 3502 Section 2
Exam #2
Version A
Spring 2017
3/23/2017

Lecturer: Arup Guha

Directions: Answer all multiple choice questions on the scantron. Each question has a single correct answer. In case of ambiguities, choose the most accurate answer. Each of these questions is worth 4 points for a correct answer. Incorrect answers and questions left blank are worth 0 points. When you finish this exam, **double check that you have bubbled in your UCFID and Exam Version on the scantron** and then hand in the scantron **ONLY**.

Consider storing a string in a linked list where each node stores a single character. The following code implements functions similar to strlen and strcmp, as well as a function to convert a regular string to its equivalent linked list representation. Several lines of the implementation have been omitted. Questions 1 - 6 will be about these missing lines.

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

typedef struct chType {
    char c;
    struct chType* next;
} chType;

chType* convert(char* word);
int len(chType* word);
int compare(chType* left, chType* right);

chType* convert(char* word) {
    chType* front = NULL;
    int i;
    for (i=strlen(word)-1; i>=0; i--) {
        chType* tmp = malloc(sizeof(chType));
        tmp->c = word[i];
        tmp->next = /*** Q1 ***/;
        front = /*** Q2 ***/;
    }
    return front;
}

int len(chType* word) {
    if (word == NULL)
        return 0;
    return /*** Q3 ***/;
}


int compare(chType* left, chType* right) {
    if (left == NULL && right == NULL)
        return /*** Q4 ***/;
    if (left == NULL)
        return -1;
    if (right == NULL)
        return 1;
    if (left->c != right->c)
        return /*** Q5 ***/;
    return /*** Q6 ***/;
}
```

1) What expression should replace /*** Q1 ***/?

a) front b) front->next c) tmp->next

d) tmp e) None of the Above

2) What expression should replace /*** Q2 ***/?

a) front b) front->next c) tmp->next

d) tmp e) None of the Above

3) What expression should replace /*** Q3 ***/?

a) len(word->next) b) 2*len(word->next) c) len(word) + 1

d) len(word->next) + 1 e) None of the Above

4) What expression should replace /*** Q4 ***/?

a) -1 b) 0 c) 1 d) left->c - right->c e) None of the Above

5) What expression should replace /*** Q5 ***/?

a) -1 b) 0 c) 1 d) left->c - right->c e) None of the Above

6) What expression should replace /*** Q6 ***/?

a) -1 b) 0 c) 1 d) left->c - right->c e) None of the Above

7) Which of the following could represent the inorder traversal of a binary search tree?

A) 9 7 3 6 8 15 12 99
B) 3 6 7 8 9 12 15 99
C) 99 12 15 9 8 7 6 3
D) All of the Above (A, B and C)
E) None of the Above

8) What is the run time of an insert inorder function on a sorted doubly linked list of n integers?

A) O(1) B) O(lg n) C) O(n) D) $O(2^n)$ E) None of the Above

9) How many structurally different binary search trees can be created which have three nodes, storing the values 1, 2 and 3?

A) 1 B) 2 C) 3 D) 4 E) None of the Above

The following code implements an insert function for a binary search tree of integers, where the height of each node is stored (distance from node to furthest leaf node in its subtree). Several lines of the implementation have been omitted. Questions 10 - 15 will be about these missing lines.

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

typedef struct bintreenode {
    int data;
    int height;
    struct bintreenode* left;
    struct bintreenode* right;
} bintreenode;

bintreenode* insert(bintreenode* root, int value);
void preorder(bintreenode* root);
void freeTree(bintreenode* root);

bintreenode* insert(bintreenode* root, int value) {
    if (root == NULL) {
        bintreenode* tmp = /*** Q10 ***/;
        tmp->data = value;
        tmp->left = NULL;
        tmp->right = NULL;
        tmp->height = /*** Q11 ***/;
        return tmp;
    }
    if (value <= root->data) {
        root->left = insert(root->left, value);
        if (/*** Q12 ***/ >= root->height)
            root->height = /*** Q13 ***/;
    }
    else {
        root->right = insert(root->right, value);
        if (/*** Q14 ***/ >= root->height)
            root->height = /*** Q15 ***/;
    }
    return root;
}
```

10) What expression should replace /*** Q10 ***/?

a) root b) malloc(sizeof(bintreenode*)) c) malloc(4)
d) new bintreenode() e) None of the Above

11) What expression should replace /*** Q11 ***/?

a) -1 b) 0 c) 1 d) root->height e) None of the Above

12) What expression should replace /*** Q12 ***/?

a) 0 b) root->height c) root->left->height

d) root->right->height e) None of the Above

13) What expression should replace /*** Q13 ***/?

a) root->left->height b) root->right->height c) root->left->height + 1

d) root->right->height+1 e) None of the Above

14) What expression should replace /*** Q14 ***/?

a) 0 b) root->height c) root->left->height

d) root->right->height e) None of the Above

15) What expression should replace /*** Q15 ***/?

a) root->left->height b) root->right->height c) root->left->height + 1

d) root->right->height+1 e) None of the Above

16) What is the value of the following postfix expression?

```
      3    4    5    +    *    3    -    1    1    +    2    *    /
```

a) 2 b) 3 c) 4 d) 5 e) None of the Above

17) What is the value of the following postfix expression?

```
      6    3    /    2    2    +    *    -
```

a) -2 b) 2 c) 24 d) Invalid Postfix Expression e) None of the Above

18) Let the sequence A of operations on a stack involve 3 pushes followed by 2 pops. Consider repeating the sequence A 20 times. If the stack was implemented using an array, what is the minimum size of the array necessary to ensure that these instructions are executed properly?

a) 20 b) 21 c) 22 d) 23 e) None of the Above

19) Which of the following class examples roughly implemented a linked list of linked lists?

A) binary tree B) stack C) queue D) Books E) Artists+CDs

Questions 20-22 involve converting the following infix expression to postfix using a stack of operators and parentheses:

(3 + 7) * ((1 + 8) - 2 * (16 / (4 - 2) - 5)) - 26

20) Right before the value 16 gets inserted into the expression, how many items are on the stack of operators and parentheses?

a) 3 b) 4 c) 5 d) 6 e) None of the Above

21) When the last close parenthesis gets processed, how many items are popped off the stack of operators and parentheses?

a) 3 b) 4 c) 5 d) 6 e) None of the Above

22) Which of the following is the equivalent postfix expression to the infix expression given above?

a) 3 7 + 1 8 + 2 16 4 2 - 5 / - * - 26 * -
b) 3 7 1 8 + + 2 16 4 2 - / 5 - * - 26 * -
c) 3 7 1 8 + + 2 16 4 2 - / 5 - * - * 26 -
d) 3 7 + 1 8 + 2 16 4 2 - / 5 - * - 26 * -
e) None of the Above

23) In class we implemented a queue using a linked list and one using an array. We tested both implementations with successive enqueues until the enqueue failed. The array implementation failed with fewer elements in the queue than the linked list version. What was our hypothesis as to why?

A) Arrays are stored inefficiently in C, compared to linked list and a single array slot takes up twice the space of the corresponding linked list node.

B) I had different programs running on my computer in the background during the two separate program runs. The programs running on my computer while I ran the array code just happened to be using more memory.

C) I ran the linked list version with a special command line flag that specified a larger allocation of memory for the execution of the program.

D) We were doubling our array size when it filled up so even though our code failed to find enough room for the doubling, there would have been room for quite a few more than one extra element. With the linked list implementation, we only added one node at a time. It only failed after nearly all the available memory was filled.

E) None of the Above

24) A short video was shown in class to introduce the breadth first search. What was the video about?

A) Trees B) Corn Mazes C) Shopping Lines
D) Rubix Cubes E) None of the Above

25) Where do seahorses live?

A) the sea B) mountaintops C) mars D) 7-11s E) igloos