BBM 201 – Data Structures - Fall 2019

Final Exam
January 22, 2020

Name: _____

Student ID Number: _____ Section:_____

Problem	Points	Grade
1	20	
2	15	
3	20	
4	15	
5	19	
6	16	
Total	105	

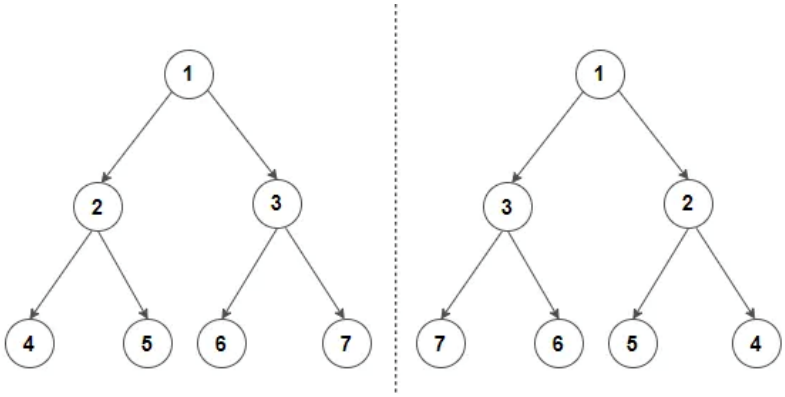
INSTRUCTIONS

- Do not open this exam booklet until you are directed to do so. Read all the instructions first.
- When the exam begins, write your name on every page of this exam booklet.
- The exam contains six multi-part problems. You have **120 minutes** to earn 105 points (5pts bonus).
- The exam booklet contains **7 pages** including this one.
- This exam is an **open book and notes exam**. You are allowed to have two pieces of **bound** books or notebooks.
- Please write your answers in the space provided on the exam paper.
- Be neat.
- Good luck!

QUESTIONS

Question 1: (18 Points)

Write a method that converts a given binary tree to its mirror as given below in the figure. The method will swap the left and right child of every node in the tree till the leaves. You can use at most one extra local variable, and no global variables. You are not allowed to do any dynamic memory allocation.




```
struct Node
{
    int data;
    struct Node* left;
    struct Node* right;
};


void mirror(struct Node* node)
{
  if (node==NULL)
    return;
  else
  {
    struct Node* temp;

    mirror(node->left);
    mirror(node->right);

    temp        = node->left;
    node->left  = node->right;
    node->right = temp;
  }
}
```

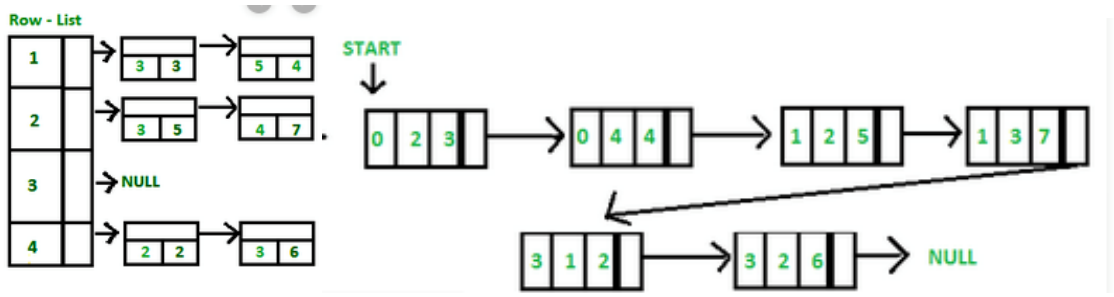
Question 2: (17 Points)

You are supposed to design an efficient data structure based on linked lists that stores a sparse matrix. For the given sparse matrix answer the below questions:

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix}$$

a) Draw your data structure. Please be neat and show all the field values of each node. Any unreadable solutions will not be graded.

Any below solution or any other reasonable solution is acceptable:



b) Give the space complexity of your structure and the time complexity of finding a value in a given (x,y) location and compare it with the array based matrix structure.

O(n)

Question 1: (15 Points) Consider a tree structure with the following definitions:

- root node: a node with no parents
- branch node: a node with a parent and non-zero children
- leaf node: a node with no children

If this tree contains **m nodes**, then answer the following accordingly:

a) (2 points) What can be the maximum number of leaves on this tree?

_____m-1_____

b) (2 points) What can be the minimum number of leaves on this tree?

_____1_____

c) (2 points) What can be the maximum number of branch nodes on this tree?

_____m-2_____

d) (5 points) If each node has at most 3 children, what can be the maximum number of leaves?

_____1+floor((m-2)/3)*2+(m-2)%3_____

or

_____m-ceiling((m-1)/3)_____

or

_____floor(m/3)+ceiling(m/3)_____

e) (4 points) Given that m>1, if each branch node has at least 2 children, what can be the maximum number of branch nodes?

_____floor((m-2)/2)_____

Question 3: (15 Points) Answer each of the following questions. Keep your answers brief.

Example) Explain a real life scenario that can better be modeled with a stack rather than a queue. Explain why a stack is better than a queue in this case.

Answer: Trying to match parenthesis in an expression can better be modeled with a stack rather than a queue. The top of the stack always stores the parenthesis that needs to be matched. But the front of a queue only stores the first parenthesis ever seen and cannot be used for matching.

a) (5 Points) Explain a real life scenario that can better be modeled with a queue rather than a stack. Explain why a queue is better than a stack in this case.

Customer request arriving at a call center.

Keyboard events received by the operating system.

Clients requesting service from a server.

In all cases the early comer has priority. Queue is FIFO and respects that. Stack is LIFO and does the opposite.

b) (5 Points) Explain a real life scenario that can better be modeled with a fixed sized array rather than a linked list. Explain why the array is better than a linked list in this case.

Storing the values of a fixed number of things is better done with a fixed size array. For example, storing the grades of a class of students ordered in a specific way. This way, the grade of any student can be accessed or modified in constant time. The linked list would be much slower to do these.

c) (5 Points) Explain a real life scenario that can better be modeled with a doubly linked list rather than a binary search tree. Explain why the doubly linked list is better than a binary search tree in this case.

You want to store and process items one by one, but their order is not important. You want the operations of insertion and removal as fast as possible without a specific order. For example, you want to store and retrieve the empty structure pointers in an application rather than freeing them. Since there is no specific order among the pointers, there is no need for the more expensive tree operations.

Question # (15 points)

Consider a hashtable implementation using separate chaining of unsorted linked lists with *N* buckets and *k* items currently in the table.

a) Which of the following is true?
   1) k < N must hold
   2) k >= N must hold
   3) Neither

b) What is the average number of items in a bucket?

c) In the worst-case, how many items could be in a single bucket?

d) Given that k > N, which of the following are true?
   1) Some buckets may be empty
   2) Some buckets may be full so that they cannot receive more items
   3) Neither

e) What is the worst case running time in terms of k and N for finding the minimum value in the table ?

f) What is the worst case run time for inserting a new item in the table ?

g) If we resize the table to a table of size 2N, what is the worst case running time in terms of k and N to put all the items in the new table?

Answers:

a) Neither

b) k/N

c) k

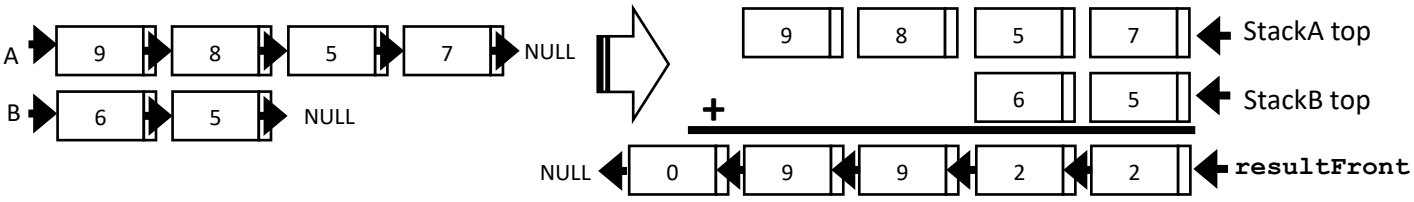
d) Some buckets may be empty

e) O(k + N). O(N) is also accepted

f) O(1)

g) O(k + N). O(N) is also accepted

Question 6: (20 points)
You are given two numbers whose digits are stored in linked lists as in the example run below. You need to add them using stacks and then store the resulting digits in a linked list. Fill in the blanks in ***int main()*** accordingly.
Hint: You need to make use of conditional expressions. Following is an example of conditional expressions: (X==Y?True:False) outputs True if X equals Y, otherwise False.
Example Run:



REQUIRED DEFINITIONS:

```
typedef struct node* node_pointer;
typedef struct node{
       int value;
       node_pointer link;
};
node_pointer resultFront;

#define MAX 1000
class Stack {
    public:
       int top;
        int a[MAX];
       Stack() { top = -1; }
       bool push(int x);
       int pop();
       bool isEmpty();
};
```

```
bool Stack::isEmpty() {
    return (top < 0);
}

int Stack::pop() {
    if (top < 0) return -1;
    else return a[top--];
}

void attach(int val, node_pointer *rear){
    node_pointer temp = new node_pointer;
    temp->value = val;
    (*rear)->link = temp;
    *rear = temp;
    (*rear)->link = NULL;
}
```

```
int main() {
    ... /* Assume that by this point the linked lists A and B have already been
    traversed and stored in StackA and StackB as in the example run above */
    node_pointer resultRear = new node_pointer;
   resultFront = resultRear;
   int e1, e2, v, carry = 0;
   while( !(StackA.isEmpty() && StackB.isEmpty()) ){
   OR while( !StackA.isEmpty() || !StackB.isEmpty() ){
   OR while( e1!=-1 || e2!=-1 ){
         e1 = StackA.pop(); e2 = StackB.pop();
         v = (e1==-1?0:e1) + (e2==-1?0:e2) + carry;
         OR v = (e1!=-1?e1:0) + (e2!=-1?e2:0) + carry;
         attach( v<10 ? v : v%10, &resultRear );
         OR attach( v%10, &resultRear );
         carry = ( v < 10 ? 0 : 1);
         OR carry = ( v >= 10 ? 1 : 0);
         OR carry = ( v / 10);
   }
   attach( carry, &resultRear );
```

```
    temp = resultFront;
    resultFront = resultFront->link;
    free(temp); return 0;
}
```