CMPSC24 Final exam
Spring 2017
6/13/2017

This exam is closed book, closed notes. You may use a one-page A4 size paper containing your nodes. Write your name on the your notes paper and all other sheets of the exam. No calculators or phones are allowed in the exam. **Write all your answers on the answer sheet. All exam sheets, notes paper and scratch paper must be turned in at the end of the exam.**

By signing your name below, you are asserting that all work on this exam is yours alone, and that you will not provide any information to anyone else taking the exam. In addition, you are agreeing that you will not discuss any part of this exam with anyone who is not currently taking the exam in this room until after the exam has been returned to you. This includes posting any information about this exam on Piazza or any other social media. Discussing any aspect of this exam with anyone outside of this room constitutes a violation of the academic integrity agreement for CMPSC24.

Signature: _____

Name (please print clearly):_____

Umail address: _____@umail.ucsb.edu

Perm number: _____

You have 3 hours to complete this exam. Work to maximize points. A hint for allocating your time: if a question is worth 10 points, spend no more than 10 minutes on it if a question is worth 20 points, spend no more than 20 minutes on it etc. If you don't know the answer to a problem, move on and come back later. Most importantly, stay calm. You can do this.

WRITE ALL YOUR ANSWERS IN THE PROVIDED ANSWER SHEET IN PEN!

DO NOT OPEN THIS EXAM UNTIL YOU ARE INSTRUCTED TO DO SO.
GOOD LUCK!

Name: _____

Part 1 [20 points] Multiple Choice: ***Select the SINGLE best answer by filling in the circles provided in your answer sheet. You must shade your selection completely as shown below:***

○ Not selected ● Selected

1. **[2 pts]** What is the worst case Big-O running time of searching for an element in a Binary Search Tree containing N elements. Assume the BST is balanced. *CHOOSE THE SINGLE BEST OPTION: the tightest bound.*

A. O(1)
B. O(N)
C. O(logN)
D. $O(N^2)$

2. **[2 pts]** Without making any assumptions about the structure of the Binary Search Tree, how does the worst case running time (Big O) of **inserting** an element into a BST with N nodes compare to that of inserting an element into a sorted linked list with N nodes? *SELECT THE SINGLE BEST ANSWER*

A. Worst case Big-O of inserting into a BST is strictly smaller than that of inserting into a sorted linked list implying that in the worst case inserting into the BST is faster
B. Worst case Big-O of inserting into a BST is the same as that of inserting into a sorted linked list
C. Worst case Big-O of inserting into a BST is strictly larger than that of inserting into a sorted linked list
D. It is not possible to compare the two data structures if we do not make assumptions about the structure (specifically the height of the BST)

3. **[2pts]** In your implementation of the ExpressionTree class in PA05, why did you implement your own copy constructor instead of using the default copy constructor? *SELECT THE SINGLE BEST ANSWER*

A. There was no particular reason. The default copy constructor would have worked as well.
B. The assignment operator would not work if we used the default copy constructor
C. The copy constructor was implemented to reduce the running time complexity of creating a new ExpressionTree
D. The default copy constructor only copies the member variables of the class and not the objects that they point to resulting in a shallow copy

4. **[2 pts]** Which of the following versions of the polynomial class implementation provides the most **space efficient** representation for polynomials with a large degree and very few terms, for example: $10x+x^{200}$. Recall that the degree of the polynomial is the largest exponent of the term with a non-zero coefficient. *SELECT THE SINGLE BEST ANSWER*

A. A static array-based implementation, where only the coefficients are stored in the array, and the exponent of each term is the index of the coefficient in the array. This version was implemented in PA03 part 1.

B. A dynamic array-based implementation, where only the coefficients are stored in the array, and the exponent of each term is the index of the coefficient in the array. The size of the array is changed dynamically at run-time. This version was implemented in PA03 part 2
C. A linked-list based implementation where the coefficient and exponent of each term in the polynomial was explicitly stored in each node of the linked-list. Terms with zero coefficients were not stored. This version was implemented in PA04
D. All the above have the same space efficiency.

5. **[2 pts]** Consider the following definition of a Node class used in the construction of a doubly linked-list:

```
template class <Item>
class Node{
      public:
         Item data;   // data element
          Node* next; //pointer to the previous node in the list
          Node* prev; //pointer to the next node in the list
          Node (const Item & d) { data = d; next = 0; prev =0;}
};
```

Which of the following statements correctly creates a Node object on the **heap** with data value set to the value of the integer **myInt**. Assume **myInt** has been declared and initialized. *SELECT THE SINGLE BEST ANSWER*

```
A. Node n(myInt);
B. Node<int> n;
C. Node<int> n(myInt);
D. Node<int>* p = new Node<int>(myInt)
```

6. **[2pts]** Which of the following is true? *SELECT THE SINGLE BEST ANSWER*

A. Different container classes usually share the same iterator class implementation
B. Iterators provide a consistent interface to access the data of a container class without knowledge of the internal representations of the class
C. Iterators can only be defined for container classes that use templates
D. None of the above

7. **[2pts]** Consider the following template function to check if two objects are equal

```
template <class Item>
bool isEqual(Item x, Item y){
      return x==y;
}
```

Which of the following conditions should be met in order to successfully compare two objects (w and z) using the above template function? *SELECT THE SINGLE BEST ANSWER*

A. The two objects must be objects of a class and not a primitive data type like (char or int)

B. The two objects must be of the same data type. Additionally if they are objects of a class, the operator == should be overloaded for objects of that class
C. The two objects must be of the same data type. No other conditions are necessary because the compiler will use the default == operator if an overloaded version of the operator is not available
D. None of the above

8. **[2pts]** Assume that you are given an implementation of the LinkedList class that implements its own copy constructor. Assume that there is an existing LinkedList object named list1. Which of the following C++ statements invokes the copy constructor of the LinkedList class? *SELECT THE SINGLE BEST ANSWER*

A. LinkedList list2 = list1;
B. LinkedList *list2 = new LinkedList;
C. Linked list2; list2 = list1;
D. None of the above

9. **[2 pts]** Which of the following operations is NOT supported by a queue data structure? *SELECT THE SINGLE BEST ANSWER*

A. Inserting an element to the rear of the queue
B. Deleting an element from the front of the queue
C. Printing the elements of the queue in sorted order
D. Select this option if all the above operations are supported by a queue

10. **[2 pts]** The C++ STL implements a balanced binary search tree (BST) as the **`set`** which provides its own forward iterator. What is the output of the following code? Assume it is embedded in a correct and otherwise complete C++ program.

```
std::set<int> mybst;
int arr[]={50, 70, 80, 10, 20, 60};
//insert the elements of the array into the BST
for(int i=0; i < 6; i++)
      mybst.insert(arr[i]);
std::set<int>::iterator it = mybst.begin();
std::set<int>::iterator en = mybst.end();
while (it != en){
      cout<<*it<< " ";
      it++;
}
```

A. 50 70 80 10 20 60
B. 60 20 10 80 70 50
C. 10 20 50 60 70 80
D. None of the above

Name: _____

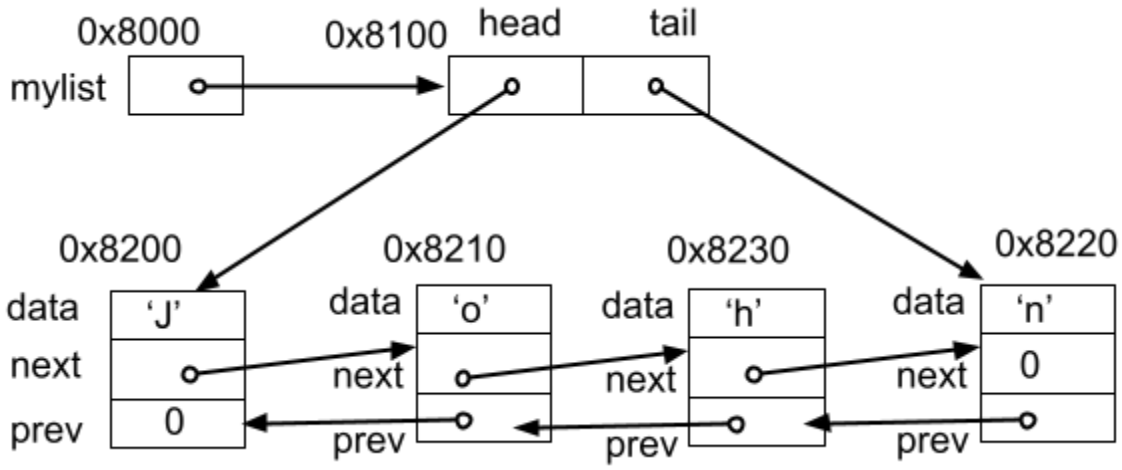
Part 2 [25 points] Linked-lists

Consider the definition of the class **Node** and class **LinkedList** which were used in the construction of a doubly linked-list. The key difference in the linked list provided below is that each node stores a char value instead of an int.

```
class Node{
  public:
    char data;  // data element
    Node* next; // pointer to the next node in the list
    Node* prev; // pointer to the previous node in the list
    Node (const char& d) { data = d; next = 0; prev =0;}
};

class LinkedList {
  public:
    LinkedList(){head = 0; tail = 0;}     // constructor

    // overloaded assignment operator makes a copy of all nodes of the
    // source list and returns the modified list
    LinkedList& operator=(const LinkedList& source);

    // destructor
    ~LinkedList();

    // Postcondition: returns true if the characters in the
    // linked-list form a palindrome
    bool isPalindrome(){
        return isPalindromeHelper(head, tail);
    }

    // inserts node with given value to the end of the list
    void insert(char value);

    // Other methods are not listed here
private:
    Node* head; // pointer to the first node in the list
    Node* tail; // pointer to the last node in the list
    // Recursively checks if the sequence of characters stored in the
    // linked list form a palindrome.
    bool isPalindromeHelper(Node* phead, Node* ptail);
};
```

Name: _____

Assume that a linked-list is already created in memory as represented by the following pointer diagram: The hex number outside each box is the starting memory location of the corresponding data object. **`mylist`** is a pointer to a **`LinkedList`** object.



1. **[5 pts]** Given the diagram on the previous page, write the output of each of the following C++ statements. Express your answer as a character (alphabet), hex or decimal as appropriate.

   a. `cout<< mylist;`
   b. `cout<< mylist->tail;`
   c. `cout<< mylist->head->next;`
   d. `cout<< mylist->head->next->data;`
   e. `cout<< mylist->tail->prev->prev->data;`

2. **[10 pts]** Implement the overloaded assignment operator, which takes a source linked list as argument and creates a duplicate of all nodes of that list in an existing list. Your assignment operator must do a deep copy of the source list and it should not have any memory leaks. You can assume that a correct implementation of the **`insert()`** method is available to you that inserts elements to the tail of the list.

3. **[10 pts]** Implement the private helper method **`isPalindromeHelper()`** to determine if the characters in the linked list form a palindrome. You must provide a RECURSIVE solution. You may assume that the linked list only contains letters of the alphabet in lower or upper case. You can use the following function to convert a character to its lower-case form:

```
char tolower(char );
```

The following sequence of characters are examples of palindromes
Abba
detarTRaTED

Name: _____

Part 3 [30 points] Binary Search Trees

Consider the definition of the class **Node** and class **BST** used in the construction of a binary search tree. Note that the provided BST stores a string in each node.

```
class Node{
  public:
    string data; // data element
    Node* left; // pointer to the left child
    Node* right; //pointer to the right child
    Node(const string& d){ data = d; left = 0; right = 0;}
};

class BST{
  public:
    BST(){root = 0; }            // constructor
    BST(const BST& source);       //copy constructor
    ~BST();                        // destructor
    //Functions to implement using an iterative solution
    //Postcondition: If node with given value does not exist, insert
    //it and return true, else return false
    bool insert(string value);


    //Postcondition: return the minimum element in the BST
    string min()const;

    //The following function should call a helper that implements a
    //recursive solution.
    void printSubsetInorder(const string value) const{
         printSubsetInorder(root, value);
    }

private:
    Node* root; // pointer to the root of the tree

    //Postcondition: Prints all values in the BST that are strictly
    //less than the given value in sorted order. You must use
    //a recursive solution
    void printSubsetInorder(Node *proot, string value) const;

};
```

Name: _____

1. **[10 pts]** Implement the **`insert()`** method of the BST class that inserts a given value that the appropriate location in the BST. The BST should not contain duplicates. If the given value is already present in the BST, the function should return false. Otherwise it should insert a new node and return true. Your solution must be ITERATIVE and not recursive.

2. **[5 pts]** Implement the **`min()`** method of the BST class that returns the minimum value stored in the BST. If the BST contains no nodes, the function should return an empty string. Your solution must be ITERATIVE and not recursive.

3. **[10 pts]** Implement the private helper method **`void printSubsetInorder(Node *proot, string value) const;`** This function takes the root node of a BST as input and prints all values in the BST **that are strictly smaller than the provided value in sorted order.** Your solution must be RECURSIVE.

4. **[10 pts]** Implement the copy constructor of the BST class. You MUST use a RECURSIVE solution. Add any helper functions that you need to the class definition and provide the implementation of the copy constructor and the helper function in your answer sheet. You don't have to rewrite the class definition.

The End

Name: _____

Scratch Paper