CPS 272 final exam – Winter 2017 – 250 points Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[1] Assume **alist** is a list of the integers {50, 15, 20, 45, 35, 10, 5}. For parts (a) and (b), display the order of elements in **alist** after calling f(); ***[6 points]***

template <typename T>

void f(list<T> &alist, T item)

{ queue<T> q;

list<T>::iterator iter = alist.begin();

while (iter != alist.end())

if (\*iter > item)

{

q.push(\*iter);

alist.erase(iter++);

}

else

iter++;

while (!q.empty())

{

alist.push\_back(q.front());

q.pop();

}

}

(a) List after calling f(alist, 30). \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(b) List after calling f(alist, 12). \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[2]. Assume an array implementation of a queue with a maximum size of 5. Initially, the queue has the characters 'A', 'B', 'C', and 'D'. Draw pictures of the initial array contents and then the contents after each queue operation. Identify the front and rear positions in each picture. ***[15 points]***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Initial View |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| q.push('E') |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| q.pop() |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| q.pop() |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| q.push('F') |  |  |  |  |  |

[3]. Use the following doubly linked list. Give the data value for the node referenced by each pointer value. ***[18 points]***



(a) p->next

(b) q->prev->prev

(c) header->next->next

(d) p->next->prev

(e) q->next->next

(f) header->prev

[4]. Trace the function f() and use it to modify elements in a linked list. ***[9 points]***

template <typename T>

void f(node<T> \* & front)

{

node<T> \*prev, \*curr, \*p;

if (front != NULL || front->next != NULL)

{

curr = front->next;

prev = front;

do

{ p = curr->next;

curr->next = prev;

if (p != NULL)

{ prev = curr;

curr = p;

}

}

while (p != NULL);

front->next = NULL;

front = curr;

}

}

Assume a linked list has the following five elements

9 -> 30 -> 17 -> 4 -> 20

front

Give the order of the elements that occur after calling f().

\_\_\_\_ -> \_\_\_\_ -> \_\_\_\_ -> \_\_\_\_ -> \_\_\_\_

front

[5]. (a) What is the runtime efficiency for inserting a node in a singly linked list? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***[3 points]***

(b) What is the runtime efficiency for deleting a node in a singly linked list? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***[3 points]***

(c) What is the runtime efficiency for locating the last node in a singly linked list? \_\_\_\_\_\_\_\_\_\_\_\_\_\_***[3 points]***

[6]. Assume the declaration

node<char> \*p, \*q, \*r;

After executing the following statements, what is the order of the data values in the resulting list? ***[3 points]***

p = new node<char>('A');

q = new node<char>('B');

r = new node<char>('C');

p->next = r;

r->next = q;

(a) A B C (b) A C B (c) C B A (d) B A C

[7]. Which of the following lists represent a possible inorder scan of a binary search tree. ***[3 points]***

(a) 7 3 8 2 9 4 11 (b) 2 3 4 7 8 9 11 (c) 11 2 9 3 8 4 7 (d) All of the above

[8]. (a) Display the binary search tree formed by entering the characters in the specified order ***[6 points]***

P, T, V, Z, S, F, C

(b) Using the tree from (a), give the NLR scan of the nodes. ***[4 points]***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(c) Using the tree from part (a), give the LNR scan of the nodes. ***[4 points]***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(d) Using the tree from part (a), give the RNL scan of the nodes. ***[4 points]***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[9]. Create a binary search tree that contains the characters: ***[3 points]***

(a) M, T, J, R, G, D (b) F, W, X, A, B, M, T, R

[10]. Trace the following tree scan function and describe its action. ***[3 points]***

template <typename T>

int treeFunc(tnode<T> \*t)

{ int n = 0, left, right;

if (t != NULL)

{

if (t->left != NULL)

n++;

if (t->right != NULL)

n++;

left = treeFunc(t->left);

right = treeFunc(t->right);

return n + left + right;

}

else

return 0;

}

(a) identifies the number of leaf nodes in the tree (b) identifies the number of nodes in the tree

(c) identifies the number of edges in the tree (d) identifies the depth of the tree.

[11]. Use the following tree that combines both operators and operands. The tree is called an expression tree.



(a) Give the postorder scan of the tree \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***[3 points]***

(b) Give the inorder scan of the tree \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ***[3 points]***

(c) Assuming the values are integers and the operators follow C++ definitions, evaluate the expression. \_\_\_\_***[3 points]***

[12]. The template class freq has data members value and count where value represents an item and count indicates the frequency of the item in a collection.

*Class Declaration*

template <typename T>

class freq

{

public:

// constructor initializes value and count with default value

// for count of 1

freq(T v, int ct = 1);

// increment the frequency by +1

void increment();

// comparison operators compare value attributes

friend bool operator< (const freq<T>& a, const freq<T>& b);

friend bool operator<= (const freq<T>& a, const freq<T>& b);

// access member functions

T getValue() const;

int getCount() const;

private:

T value;

int count;

};

Parts (a) - (c) explore elements of a program that declares a set of freq objects and uses elements from an array of integers to insert and update the freq objects in the set.

int arr[] = {6, 3, 3, 9, 4, 9, 6, 9, 3};

(a) Declare s to be a set of freq objects whose values are integers. ***[3 points]***

(i) set<freq, int> s; (ii) set<freq<int> > s;

(iii) freq<set<int> > s; (iv) set<freq> s<int>;

(b) Complete the loop that inserts a new freq object in the set for the first occurrence of a value in the array or updates the count of an existing freq object if the array value is a duplicate. ***[6 points]***

for (int i = 0; i < 9; i++)

{

if ((sIter = s.find(arr[i])) == s.end())

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ // select the code

(i) s<freq<int> >.insert(arr[i]);

(ii) s.insert(freq(arr[i]));

(iii) s.insert(arr[i]);

(iv) s.insert(freq<int>(arr[i]));

else

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ // select the code

(i) (\*sIter).increment();

(ii) \*(sIter).increment();

(iii) s.increment(\*sIter);

(iv) s.increment(sIter);

(c) The loop scans the set, outputting the value and the count components of each freq object in the format value (count). Give the resulting listing. ***[3 points]***

for (sIter = s.begin(); sIter != s.end(); sIter++)

cout << (\*sIter).getValue() << "(" << (\*sIter).getCount() << ") ";

Output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(d) Rewrite the if..else statement using the insert() member function of the set class. Use the following declaration of an iterator/bool pair. ***[3 points]***

pair<set<freq<int> >::iterator, bool> p;

[13]. Let map m store key value pairs where the key component is an integer item and the value component is the frequency of the value in a collection. The integer array arr defines a list of nine elements.

int arr[] = {6, 3, 3, 9, 4, 9, 6, 9, 3};

map<int, int> m;

Complete the loop that scans the elements in the array and adds a new entry in the map for the first occurrence of a value and increments the value component for a duplicate value. ***[3 points]***

for (int i = 0; i < 9; i++)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ // select from choices (i) through (iv)

(i) m(arr[i]) = (m(arr[i]) > 1) : m(arr[i])++ ? 1;

(ii) m[arr[i]] += 1;

(iii) m[arr[i]++];

(iv) None of the above

[14]. The following is a listing of the miniMap class index operator implementation. Fill-in the missing portions. For your reference, here is a listing of the miniMap private section. ***[12 points]***

private:

// miniMap implemented using an stree of key-value pairs

stree<miniPair<const Key,T> > t;

template <typename Key, typename T>

T& miniMap<Key,T>::operator[] (const Key& key)

{

// build a miniPair object consisting of key

// and the default value of type T

value\_type tmp(\_\_\_\_\_\_\_, \_\_\_\_\_\_\_);

// will point to a pair in the map

iterator iter;

// try to insert tmp into the map. the iterator

// component of the pair returned by t.insert()

// points at either the newly created key/value

// pair or a pair already in the map. return a

// reference to the value in the pair

iter = t.insert(tmp).\_\_\_\_\_\_\_\_;

return (\*iter).\_\_\_\_\_\_\_\_;

}

[15]. What is the output of this program? ***[3 points]***

#include <iostream>

#include <fstream>

#include <string>

#include <map>

using namespace std;

void main()

{ ifstream collectionData;

string name;

int dollars;

map<string,int> canvassers;

map<string,int>::iterator iter;

collectionData.open("canvas.txt");

while (true)

{ collectionData >> name >> dollars;

if (!collectionData)

break;

canvassers[name] += dollars;

}

for (iter=canvassers.begin(); iter != canvassers.end(); iter++)

cout << (\*iter).first << " " << (\*iter).second << endl;

}

File "canvas.txt"

Ted 10

Alice 20

Bob 50

Ted 23

Joan 60

Alice 20

Bob 30

Alice 70

Ted 80

Joan 50

Run:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[16]. The accompanying 2 - 3 - 4 tree results from inserting the sequence of elements

{38, 16, 29, 50, 60, 20, 45, 70, 52, 10, 5, 65}

Draw the corresponding red-black tree. ***[3 points]***



[17]. Use the following sequence of integer values.

8, 1, 5, 3, 16, 18, 7, 33, 9, 15, 22, 10

Build the 2 - 3 - 4 tree. ***[3 points]***

[18]. Use the sequence from Question 17 above to build the red-black tree. ***[6 points]***

[19] Use the integer hash function hf(x) = x and table size 11. Using chaining with separate lists, show the

location in the hash table for each integer value in the following sequence. ***[3 points]***

5, 19, 43, 38, 63, 96, 44, 65

|  |  |
| --- | --- |
|  | ht |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |

[20] The declaration creates a multiset ms of character elements. Assume a loop scans the elements in the string

"Mississippi" and inserts them in the multiset.

string str = "mississippi";

multiset<char> ms;

In parts (a) and (b), give the resulting output.

(a) cout << ms.size(); // Output: \_\_\_\_\_\_\_\_\_\_\_\_\_***[3 points]***

(b) // Output the elements in the multiset

writeContainer(ms.begin(), ms.end());

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***[3 points]***

[21] Which STL typedef creates the identifier value\_type for a map object: ***[3 points]***

(i) typedef map<Key, T> value\_type

(ii) typedef map.value\_type(Key, T)

(iii) typedef pair<Key, T> value\_type

(iv) typedef map<pair<Key, T> > value\_type

[22] Assume a map type consists of an int for the key component and a string for the value component. Define a typedef statement that equates the identifier entry with a map value\_type. ***[3 points]***

[23] What is the output of the program that uses a map with int and string components for the key\_value pair ?

map<int,string> m;

map<int,string>::iterator iter;

string arr[5] = {"one", "two", "three", "four", "five"};

int j;

for (j=0;j < 5;j++)

m[j] = arr[j];

m[2] += " o'clock";

iter = m.find(3);

cout << (\*iter).second << endl;

for ( iter = m.begin(); iter != m.end() ; iter++ )

cout << (\*iter).first << " " << (\*iter).second << endl;

Output:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ***[3 points]***

[24] Hashing is an \_\_\_\_\_\_ algorithm. ***[3 points]***

(a) O(log2n) (b) O(n2) (c) O(n) (d) O(1)

[25] Determine output1 and output2.

template <typename T>

class up

{

public:

up(const T& b = T()): base(b){}

T operator() (const T& x){ return x - base; }

private:

T base;

};

up<int> obj(3);

cout << obj(5) << endl; // output 1: \_\_\_\_\_\_\_\_\_\_\_\_\_\_***[3 points]***

cout << up<double>() (2.718) << endl; // output 2: \_\_\_\_\_\_\_\_\_\_\_\_\_\_***[3 points]***

[26] Assume that a hash function has the following characteristics.

Keys 203, 426, and 561 hash to 5

Keys 987 and 316 hash to 3

Key 736 hashes to 2

Key 124 hashes to 0

Assume that insertions are done in order 987, 203, 736, 316, 426, 561, 124

(a) Indicate the position of the data if chaining with m = 7 is used to resolve collisions. ***[3 points]***

|  |  |
| --- | --- |
| ht | |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |

(b) Which element(s) require the largest number of probes to locate it in the table? \_\_\_\_\_\_\_ ***[3 points]***

(c) Which element(s) can be accessed with a single probe?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ***[3 points]***

[27] ***[70 points]*** Programming part: There are two techniques to resolve collisions using hashing, chaining technique and Linear Probe Addressing. In the classroom, we implemented the chaining technique to resolve collisions. You need to implement the other technique to resolve collisions using hashing. Create the classes you need and create a main program to show that it is working.