

Name: key

PLEASE PRINT CLEARLY

ECE x70 Exam No. 3 (100pts. - 25% of the final grade)

General Remarks

This is in-class one-hour-long exam. You can use your notes, textbook, and any existing Web-based resources. You can use a lab or your own computer but must not use a cell phone. You must not communicate with other people or post the questions on an Internet forum. Provide a concise answer and to the point for maximum credit. Answers that are too long take too much time and may indicate that the author is unable to rank the importance of facts.

DL: __ ERR: __ PTS: __

DL – exam difficulty level (adjustment), ERR – exam errors, PTS – exam points.

Problem 1 (25pts.) – Numerical complexity and operations in our examples

Analyze the code and show what it does by showing what is stored in each variable when the code is run. Use ‘-’ to indicate that the data container is empty and nothing valid is stored at a time. STL library is used.

```
list<int> L; list<int>::iterator IT;
```

Consider the following fragment of code:

```
L.push_back(1); L.push_front(2); L.push_front(3); L.push_back(4);
```

The list contains: 3 2 1 4

Now assume that the list contains the following data: 1 2 3 4

```
IT = L.begin(); ++IT; IT++; L.insert(IT, 5); ++IT; ++IT; L.insert(IT, 6);
```

The list contains: 1 2 5 3 4 6

Now assume that the list contains the following data: 1 2 3 4 5 6

```
IT = L.begin(); IT++; ++IT; IT=L.erase(IT); L.erase(IT);
```

The list contains: 1 2 3 6

Now assume that the list contains the following data: 1 2 3 4

```
for(auto II=L.rbegin(); II!=L.rend(); ++II) cout << ' ' << *II;
```

What is printed: 4 3 2 1

// each missing, extra or misplaced number costs 1p., except the reverse of the last answer costs 4p.

Question 2A (5p.) – Numerical Complexity – Big O notation

Dr. Wangs's crazyfly robot armada on average maps BECC Control Lab in about 220 seconds before returning to their dock station. After moving it to a basketball court in the Markin Center that is approximately three times larger in each direction it was determined that the armada needs about 2820 seconds to perform the same operation. What is the most likely the numerical complexity of that mapping process with respect to the room linear dimension as N ? Docking flying robots takes always the same extra time that can be almost neglected and it is included in the measured time. (Note: the story is made up, do not try to reason the complexity based on the task) Circle the closest answer:

2^n n^5 n^4 n^3 $n^2 \cdot \text{sqrt}(n)$ $n^2 \cdot \log(n)$ n^2 $n \cdot \text{sqrt}(n)$ $n \cdot \log(n)$ $n \cdot \text{sqrt}(n)$ $\log(n)$ 1 0

Question 2B (5p.) – Numerical Complexity – Time estimation

Dr. Wangs's crazyfly robot armada on average maps BECC Control Lab in about 220 seconds before returning to their dock station. Assuming that the numerical complexity of mapping rooms on the same floor in respect to the room linear dimensions is $O(n^2)$ how much time it would take to map BECC that is approximately four times larger? Circle the closest answer:

more 100,000 50,000 30,000 20,000 10,000 5,000 3,000 2,000 1,000 300 200 100

Question 2C (15p.) – Numerical Complexity and STL Library functionality

Based on your knowledge of the data container implementations of `vector<T>` (SimpleVector), `list<T>` (homework 12), and `deque` (CircularBuffer) circle the closest numerical complexity for the member functions of these containers. Assume N is volume of data held in a container.

`V.insert(V.end(), x)` where V is `vector<T>` 1 n n^2 more

`V.insert(V.begin(), x)` where V is `vector<T>` 1 n n^2 more

`V.clear()` where V is `vector<T>` 1 n n^2 more

`L.insert(L.end(), x)` where L is `list<T>` 1 n n^2 more

`L.insert(L.begin(), x)` where L is `list<T>` 1 n n^2 more

`L.clear()` where L is `list<T>` 1 n n^2 more

// Note: these particular questions should not be semester-dependent

// Questions referring to homework or class examples may change answers

Problem 3 (25pts.) – Understanding linked-list-alike structures

Analyze the implementation of the two dimensional linked-list-alike data structure and then implement requested functions according to the comments included with each of them.

//ILLUSTRATION OF THE IDEA:

address			link		
	U				
L	X	R			
	D				

list		
start		
size		

iterator		
current		

4000		
	0	
0	11	4040
	4200	

4040		
	0	
4000	12	4080
	4360	

4080		
	0	
4040	13	4120
	4280	

4120		
	0	
4080	14	4160
	4320	

4160		
	0	
4120	15	0
	4360	

4200		
	4000	
0	21	4240
	4400	

4240		
	4360	
4200	placeholder	4280
	4640	

4280		
	4080	
4240	23	4320
	4480	

4320		
	4120	
4280	24	4360
	4520	

4360		
	4160	
4320	25	0
	4560	

4400		
	4160	
0	31	4480
	4600	

example of an already erased item

4480		
	4280	
4400	33	4520
	4680	

4520		
	4320	
4480	34	4560
	4720	

4560		
	4360	
4520	35	0
	4760	

4600		
	4160	
0	41	4640
	0	

4640		
	4240	
4600	42	4680
	0	

4680		
	4480	
4640	43	4720
	0	

4720		
	4520	
4680	44	0
	0	

4760		
	4560	
4720	45	0
	0	

```
//CODE:
```

```
class myLink {
friend class myList;
friend class myIterator;
    myLink *U, *D, *L, *R;
    int    X;
public: void erase(...);
};

class myList {
    myLink *start;
};

class myIterator {
    friend class myList;
    myLink *current;
public:
    myIterator(myLink* x) : myLink(x) {}
    void up() const;
};
```

```
// remove the link pointed by the iterator from the grid and deallocate its memory
// note that you may be erasing an edge node. you are not erasing placeholder node
void myList::erase(const myIterator& it) { // answer worth 20 points
```

```
if ( it.current->U ) it.current->U->D = it.current->D;
```

```
if ( it.current->D ) it.current->D->U = it.current->U;
```

```
if ( it.current->L ) it.current->L->R = it.current->R;
```

```
if ( it.current->R ) it.current->R->L = it.current->L;
```

```
delete it.current;
```

```
// each error within a line of code, or extra line of code costs 1 point
// consistently the same error across all commands costs 4 points
```

```

// make an iterator point to one element up (assume the link up is valid)
void myIterator::up() { // answer worth 5 points

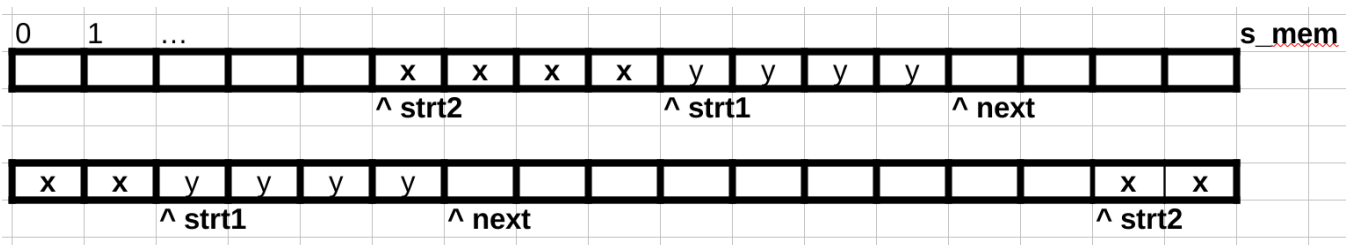
```

```
current = current->U;
```

```
}
// Note: problem introduced previously in class during review,
// questions for more functionality of this structure to come in future tests
```

Problem 4 (25pts.) – Algorithm implementations in code

Complete implementation of the code to compute a short average of data sequence as indicated by xxxx in the example figure. The data is stored in a circular buffer like one in the recent homework assignment. For full credit add only as little additional variables as absolutely necessary. Utilization of the already defined variables can be determined by analyzing the code of the provided storeNext(..) function. Everything is provided in a class template notation. Assume that, unlike in the homework, you cannot initialize sum using value of 0 (i.e., “= 0;”) assignment. The actual values of s_av2 and s_av1 may vary from the figure below.



```
template <typename Tdata, size_t s_av2, size_t s_av1, size_t s_mem>
class TrendPredictor {
private:
    size_t strt2, strt1, next, size _____ ; // blanks are optional
    Tdata buffer[s_mem];
public:
    TrendPredictor():strt2(0),strt1(0),next(0),size(0) _____ {}
    bool ready() const { return(size>=s_av2); }
    void storeNext(const T& x) {
        buffer[next] = x;
        ++next; if (next==s_mem) next=0;
        if (size<s_mem) ++size;
        if (size>s_av1) { ++strt1; if (strt1==s_mem) strt1=0; }
        if (size>s_av2) { ++strt2; if (strt2==s_mem) strt2=0; }
    }
    T average_2() const {
        T sum; // for full credit try not to use: sum=0;
        sum = buffer[strt2]; // A - sum initialization
        size_t ndx = (strt2 + 1) % s_mem; // B - start correctly
        while ( ndx != strt1 ) { // C - correct stop
            sum = sum + buffer[ndx]; // D - keep adding up
            ndx = (ndx + 1) % s_mem; // E - advancing correctly
        }
        sum = sum / ( av1 - av2 ); // F - finish computing average
        return(sum);
    }
};
// 4 points each A B C D E F, 2 points for "=0", 1 point extras
// Note: with -2p for =0 this collapses to short average with strt1 as next
// A task to do in your homework assignment - assesses your work on homework
```