CE 013312 Midterm Exam

You will have 2 hours to complete the following exam. The exam is open-book, open-note, and open-internet, however, you cannot use the internet to communicate with any other person. This includes any messaging, social media, or shared document applications.

You should attempt to answer every question. If you believe that a question is not clear, state an assumption and attempt to answer it. If you believe there are multiple possible correct answers select the one that you think is the best.

Chain::RemoveLast()

The following questions consider the implementation shown below of a new RemoveLast() method for the Chain() linked-list class discussed in class:

```
template <class T>
T Chain<T>::RemoveLast() {
    if (first == NULL)
        throw "Can't remove element from an empty list";
    ChainNode<T> *node, *prev = NULL;
    for (node = first; node->link != NULL; node = node->link)
        prev = node;
    T nodedata = node->data;
    prev->link = NULL;
    delete node;
    return nodedata;
}
```

1. Which of the following best describes the purpose of the variable "prev" in the given segment?

1. A copy of the data field from the previous ChainNode
2. Either a pointer to the first or last node of the Chain
3. A pointer to node that will be removed
4. A pointer to the ChainNode pointing to the current node unless the current node is the first
5. A pointer to the node that will be pointed to by first when last node is removed

정답: 4

2. The given method is incorrect for which one of the following cases?

1. the list is empty
2. the list has only one node
3. the list has only two nodes
4. the node is deleted before the nodedata is returned
5. When two nodes have the same data values

✎ 정답: 2

3. Given a list *L1* and a second empty list *L2*, describe the resulting lists after executing the following code fragment within a Chain() class member function:

```
while (L1.first != NULL)
    L2.InsertFirst(L1.RemoveFirst());
```

✎ 정답

The elements of L1 are copied to L2 in reverse order, and L1 is empty

4. Suppose that we modified our `Chain()` linked list implementation to include ChainNode pointers to both the first and last ChainNode. When the list is empty, both *first* and *last* would be NULL, when the list has only a single element, both *first* and *last* would both point to it,
otherwise, *first* and *last* would point to the appropriate nodes. As a result of this addition it will be necessary to add code to methods that possibly change last. Select below all methods that need to be modified.

1. InsertFirst()
2. InsertLast()
3. InsertOrdered()
4. RemoveFirst()
5. RemoveElement()
6. Reverse()

✎ 정답: 1 2 3 4 5 6

5. Adding a last pointer to our Chain() linked-list implementation allows us to implement some ADT operations in *O(1)* time that were previously *O(N)*,

where *N* is the length of the list. Select below all methods can be sped up by including last.

1. InsertFirst()
2. InsertLast()
3. InsertOrdered()
4. RemoveFirst()
5. RemoveElement()
6. Reverse()

✎ 정답: 2

Recursive Traversals

Linked lists can also be traversed using recursive functions. Consider the following Search() method for our Chain() linked list implementation which returns the address of a the first Node with data matching the target argument:

```
template <class T>
ChainNode<T> *Chain<T>::Search(T target, ChainNode<T> *node) {
    if (node == NULL)
        node = first;
    if (node->data == target)
        return node;
    else if (node->link == NULL)
        return NULL;
    else
        return Search(target, node->link);
}
```

6. In the space below, write a single declaration line for the given Search() method that you would include in the Chain.h header file. Make sure that your declaration sets an appropriate default value for the node argument.

✎ 정답

```
ChainNode<T>* Search(T target, ChainNode<T> *node = NULL);
```

7. Which of the following occurs if the target value is not present in the list?

1. A NULL pointer is returned

2. The recursion does not end
3. An exception is generated
4. The address of the first Node is returned
5. The address of the last Node is returned

✎ 정답: 1

8. Is the given code for Search() correct? If it is not, describe any case where it gives wrong or unexpected results, generates an error, or it does not terminate. If it is, explain why it is correct. You should assume that the link-list is correctly constructed and that the initial call to Search is with valid arguments.

✎ 정답

It generates an error when searching and empty list (first = NULL). A test for if (first == NULL) needs to be added at the beginning and it should either generate an exception or return NULL.

Node Swap

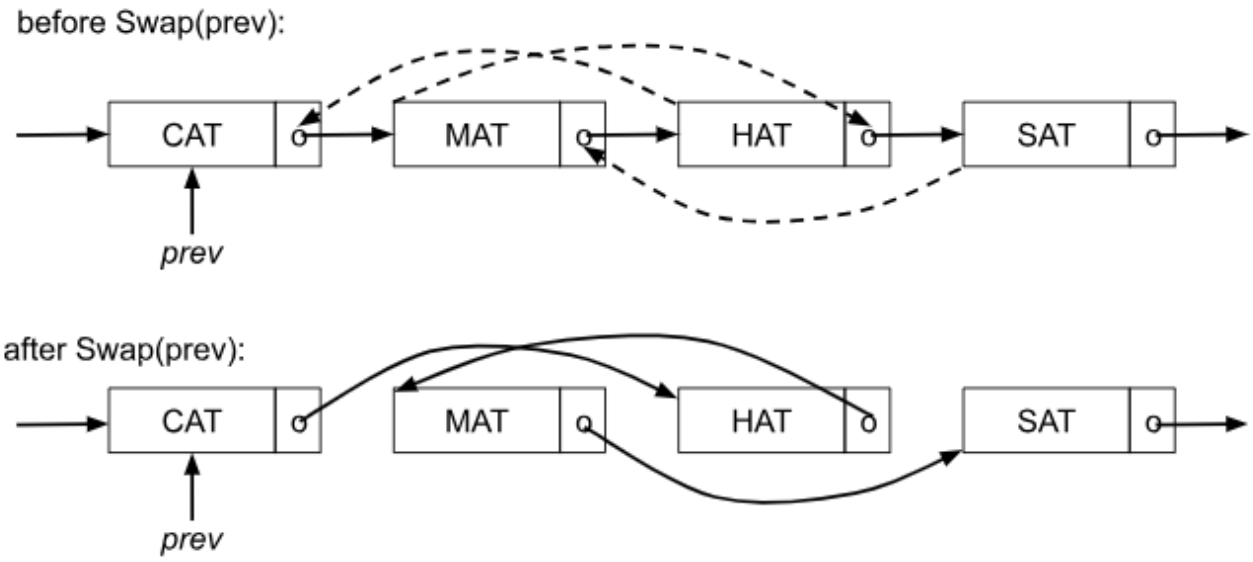
In this section we will consider adding a `void Swap(ChainNode<T> *prev)` method to our `Chain()` linked list implementation that swaps the node after `*prev (prev->link)` with the node following `prev->link (prev->link->link)`, without moving "data".

The Swap(prev) method performs the following link updates:

```
ChainNode<T> nxt = prev->link; // pointer to MAT
ChainNode<T> nxtnxt = nxt->link; // pointer to HAT
ChainNode<T> nxtnxtnxt = nxtnxt->link; // pointer to SAT
prev->link->link->link = nxt; // update HAT's link
prev->link->link = nxtnxtnxt; // update MAT's link
prev->link = nxtnxt; // update CAT's link
```

As depicted below:

Chain links before and after the `Swap(prev)` method



9. Which of the following edge cases must Swap(prev) handle as special cases? (Check all that apply)

1. first == NULL, the list is empty
2. first->link == NULL, the list has only one node
3. prev->link == NULL, prev is the last node
4. prev->link->link == NULL, prev points to the node before the last one
5. prev == NULL, swap the first two nodes

✎ 정답: 1 2 3 4 5

10. Write an implementation of the *Swap(prev)* method in the space provided.

✎ 정답 code

```
template <class T>
void Chain<T>::Swap(ChainNode<T> *prev) {
    if (first == NULL)
        throw "Can't swap nodes in an empty list";
    if (first->link == NULL)
        throw "Can't swap nodes in a list of one";
    ChainNode<T> *nxt = (prev == NULL) ? first : prev->link;
    if (nxt == NULL)
        throw "Can't swap node after the end";
    ChainNode<T> *nxtnxt = nxt->link;
    if (nxtnxt == NULL)
        throw "Can't swap the node before the end";
    ChainNode<T> *nxtnxtnxt = nxtnxt->link;
    if (prev != NULL) {
        prev->link->link->link = nxt;
        prev->link->link = nxtnxtnxt;
```

```
            prev->link = nxtnxt;
        } else {
            first->link->link = nxt;
            first->link = nxtnxtnxt;
            first = nxtnxt;
        }
    }
```

11. Of the two sorting methods discussed in class SelectionSort() and SwapSort(), which is best suited to take advantage of our Chain Class's Swap() method? Explain your answer.

✎ 정답

It can be used to implement SwapSort() since it exchanges adjacent nodes, whereas SelectioSort() swaps the minimum item in the unsorted part of the list (the end). It will, however, need to use a lagging "prev" pointer since both swap targets follow the *node argument.

12. Which of the following, when included in a class's header (.h) file exposes implementation details of the Abstract Datatype.

1. private member variables
2. in-line construtor code
3. Data-only classes declared to be friends of the given class
4. An inline function that accesses private member variables rather than getters and setters
5. All of the above

✎ 정답: 5

Sparse Toeplitz Matrices

The following ADT describes a subtype of a SparseMatrix called a Toeplitz matrix, and is implemented entirely within the "Toeplitz.h" header file similar to how the Stack and Queue classes were implemented by inheriting from the Chain object in Lecture 9. A Toeplitz matrix has constant values along each of its diagonals. Toeplitz matrices are widely used in machine learning applications to perform the "convolutions" of a Convolutional Neural Network (CNN).

```
#include "SparseMatrix.h"
using namespace std;

#ifndef TOEPLITZ_H
#define TOEPLITZ_H
class Toeplitz : public SparseMatrix
{
public:
    Toeplitz(int row, int col, int bandwidth = 0, int diag[] = NULL)
    : SparseMatrix(row, col, (2*bandwidth+1)*row)
    // returns a Toeplitz matrix with the given bandwidth
    // initialized by T[i,j] = diag[bandwidth+j-i] where
    // values has 2*bandwidth+1 entries. Defaults to an
    // Identity matrix (T[i,i] = 1).
    {
        int minc, maxc;
        for (int r = 0; r < row; r++) {
            minc = r - bandwidth;
            minc = (minc < 0) ? 0 : minc;
            maxc = r + bandwidth;
            maxc = (maxc >= col) ? col-1 : maxc;
            for (int c = minc; c <= maxc; c++) {
                int v = (diag != NULL) ? diag[bandwidth + c - r] : 1;
                setvalue(r,c,v);
            }
        }
    }
};

#endif
```

13. Which of the following statements about the given Toeplitz constructor is false?

1. SMArray values are filled in the expected row-col order
2. The resulting Toeplitz matrix is sparse for values of bandwidth less than `min(row, col)`
3. The transpose of a Toeplitz matrix is a Toeplitz matrix
4. Adding two Toeplitz matrices produces a Toeplitz matrix
5. The running time of the Toeplitz constructor is `O(bandwidth * row)`

✎ 정답: 2

14. Even when the elements of diag are all non-zero, the given implementation of the Toeplitz matrix constructs a sparse matrix with extra capacity. In other words, there are fewer than `(2*bandwidth+1)*row` non-zero terms if bandwidth

is larger than zero. In the space below give a tighter expression for setting the capacity of the SparseMatrix( ) constructor.

정답

```
row + bandwidth * (2 * row - bandwidth - 1)
```

15. In the space provided discuss any disadvantages of using the SparseMatrix::setvalue() method in the inner loop of the Toeplitz() constructor caused by not having access to the private class variables.

정답

Our method fills the values of the Toeplitz smArray in the correct order (increasing row, increasing col). However, the setvalue() method must scan through the smArray every time only to find that it inserts each new value into the last position and increases terms by 1.