FSU COP 4530 / CGS 5425 (Fall 2005) Sec 1-2
Data Structures, Algorithms, and Generic Programming

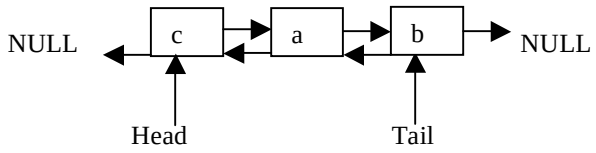
Midterm: Max points: 100, Time: 75 minutes

First Name: _____ Last Name: _____

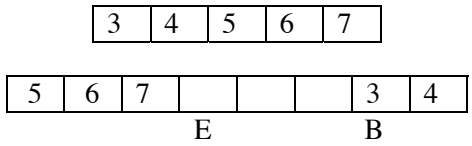
This is a closed book examination.

1. (a) (15 points) Draw the state of a self-organizing doubly linked list using the `transpose` method, after the following operations. Show the links too.

insert(a); insert(b), insert(c), search(a), search(a), search(a), search(c), search(c),



(b) (20 points) Let us perform the following operations on a queue: `push(1)`, `push(2)`, `push(3)`, `pop()`, `pop()`, `push(4)`, `push(5)`, `push(6)`, `push(7)`. (i) Draw a figure to show the state of the queue. If the underlying data structure were a deque, with initial capacity 4, then show the final state of the deque too. Mention which deque operation you used to `push`, and which operation you used to `pop`.



push = push_back => pop = pop_front
push = push_front => pop = pop_back

2. (a) (10 points) Let the time complexity of an algorithm be $20\ n^2 + 2\ n^2 \log n - 2n + 1$. Give the asymptotic time complexity in big-Oh notation.

$O(n^2 \log n)$

(b) (10 points) How many times is the statement with *cout* << ... executed, in the following pseudocode? Derive the exact number, showing all steps.

```
for i = 1 to n
  for j = 1 to i
    for k = 1 to n
```

cout << i << j << k << endl;

$\Sigma^{n}_{i=1} \Sigma^{i}_{j=1} \Sigma^{n}_{k=1} 1 = n \Sigma^{n}_{i=1} \Sigma^{i}_{j=1} 1 = n \Sigma^{n}_{i=1} i = n^2(n+1)/2$

3. (20 points) Let DLL be a self-organizing doubly linked list class using a `reversetranspose` method, with Node *head and Node *tail pointing to the head and the tail of the list, respectively. In the reversetranspose method, when we search for a node and find it, we will exchange its position with that of the next node in the list (if such a node exists), in contrast to the transpose method, which exchanges position with the previous node. Implement a member function called `reversetranspose`, which is given as argument a pointer to a node that has been found. This function implements the operation that exchanges positions. Please ensure that the list is in a valid state after this operation, and that your code will not seg-fault. You may assume that the argument points to a valid node in the linked list. You may also assume that the linked list stores only `int`s; consequently, your code need not be templated.

```
void reversetranspose(Node *N){

   if(N == tail)
       return;

   N->next->prev = N->prev;
   if(N != head)
       N->prev->next = n->next;
   else
       head = N-> next;

   if(N-next->next)
       N->next->next->prev = N;
   else
       tail = N;

   N->prev = N->next;
   N->next = N->next->next;
   N->prev->next = N;

   return;

   }
```

4. (25 points) This question primarily tests your knowledge of the following: (i) writing and using template classes, (ii) using header file guards, (iii) reading input from a file, (iv) implementing features of a stack, and (v) compiling. You will write a **complete program**, consisting of more than one file. Give the file name for each file, above the code that file contains.

First, implement a templated class called Stack, which implements a generic stack using an STL deque. (Unlike with the STL stack, the underlying container type will always be a deque.) The following features of a stack should be implemented: (i) `void pop()`, (ii) `void push(T &)`, (iii) `T &top()`, and (iv) `bool empty()`. These functions will call appropriate functions of a deque. Use this class in a main program, which reads integers from a file called file.in. file.in (which you don't need to write) contains one integer on each line. The main program will read each integer, and using the stack, output the values read in reverse order.

Compilation command: g++ main.cpp

Code:

header.h

```
#ifndef HEADER_H
#define HEADER_H
#include<deque>
using namespace std;

template <class T> class Stack
{
  deque<T> D;

  public:

  void pop(){D.pop_back();}
  void push(T &t){D.push_back(t);}
  T &top(){return D.back();}
  bool empty(){return D.empty();}
};

#endif
```

main.cpp

```
#include<iostream>
#include<fstream>
#include “header.h”

using namespace std;

int main()
{
  ifstream is(“file.in”);
  Stack<int> S;
  int temp;

  while(is >> temp)
     S.push(temp);

  while(!S.empty())
  {
     temp = S.top();
     cout << temp << endl;
     S.pop();
   }

  return 0;
}
```