FINAL EXAMINATION – DECEMBER, 2003
CSC 112
NICHOLAS R. HOWE

This is a closed-book exam. You may use two double-sided 8.5x11 sheets of notes.

All answers to this exam should be written in your exam booklet(s). Start with the questions that you know how to do, and try not to spend too long on any one question. Partial credit will be granted where appropriate. You will have two hours and twenty minutes. Good luck!

Data Structure Selection

1. (10 points) Select the most appropriate data structure to help solve the following problems. Be as specific as possible; if you can supply a more specific name than a general data structure ("binary search tree" vs. "tree"), then you should do so.

a. You want to write a program that will store and compute the value of boolean expressions using the binary boolean operators OR and AND, such as ((*P* AND *Q*) OR (*P* AND *R*)).

b. An operating system must keep track of the set of active processes, which are stored in no particular order. From time to time, it must check each of the processes to update its state. Sometimes a process will terminate, in which case its record will be eliminated. Other times a process will split in two, in which case an identical record must be created beside the first.

c. A *reversing buffer* works as follows: a sequence of characters is stored one at a time in the buffer. When finished, the characters may be retrieved from the buffer in the opposite order from which they were stored.

d. Internet routers receive incoming packets and quickly forward them on to their destinations. If they get a heavy burst of incoming packets, they temporarily store them and process them in the order in which they were received.

e. A geographic system must efficiently locate data associated with two-dimensional coordinate keys (i.e., the location where the measurement was taken).

C++ Program Design

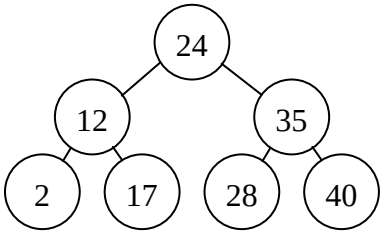
2. (10 points) Write a short (several paragraph) response to each of the questions below.

a. Discuss *encapsulation*, giving examples of mechanisms within C++ that help to achieve it. Explain how encapsulation helps make a class more abstract, using a specific example.

b. Discuss the advantages and disadvantages of using exceptions, in comparison with other methods of handling errors.

Trees

3. (10 points) Given the binary search tree shown below, draw the data structure that would result from the following operations. (For each part of the problem, assume that you start with the original tree.)



a. Insert 19.

b. Insert 33.

c. Delete 12, followed by a left copy if necessary.

d. Delete 28, followed by a right merge if necessary.

e. Perform a right rotation of the root.

Heaps

4. (12 points) The diagram below shows the contents of a heap at a particular point in time, stored within an array as discussed in class. Show the result of each operation listed below, by giving the value returned (if any) and drawing the state of the data structure after the operation is complete. Assume that each operation is applied successively, so that part (b) uses the result of part (a), etc. (Hint: It may help to draw the structure as a tree so you can see what is happening. However, in your answer you must show the data as an array.)

	60	48	57	3	41	50										

a. *Insert(77)*

b. *Insert(5)*

c. *RemoveRoot()*

d. *RemoveRoot()*

e. *Insert(26)*

STL

5. (12 points) The program below was written using the `DL_IntList` class developed for this course. Rewrite it so that it uses the templated `list` class and iterators from the Standard Template Library to accomplish the same effect.

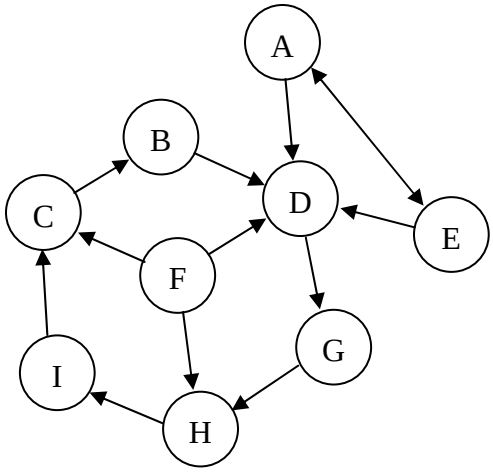
```
#include "DL_IntList.h"
#include <iostream>
using namespace std;

// Return the position of the smallest element in a list
// R/O ls:  list to look in
DL_IntListNode *minpos(const DL_IntList &ls) {
  DL_IntListNode *result = ls.getHead();
  DL_IntListNode *item;
  for (item = ls.getHead(); item != 0; item = item->getNext()) {
    if (result->getValue() > item->getValue()) {
      result = item;
    }
  }
  return result;
}

int main() {
  DL_IntList ls;

  ls.insertAtTail(5);
  ls.insertAtTail(3);
  ls.insertAtTail(-2);
  ls.insertAtTail(0);
  ls.insertAtTail(-2);
  cout << minpos(ls)->getValue() << endl;
}
```

Graphs

6. (12 points) Identify the order in which vertices would be visited for the specified traversal of the **directed** graph below. Also identify any vertices that are unreachable from the starting vertex. Where there is a choice of more than one vertex, choose the one that is first in alphabetical order.

a. Breadth-first traversal starting at F.

b. Depth-first traversal starting at A.



Stacks and Queues

7. (12 points) For the following sequence of operations, give the result returned by each `pop()` and `out()` operation, in order, and draw the final state of the resulting data structure with links shown. Assume that both stacks and queues are implemented as described in class, and that the data structures are initially empty.

a. Stack: push(7); push(6); pop(); push(4); pop(); push(11); push(7); push(8); pop(); pop().

b. Queue: in(7); in(6); out(); in(4); out(); in(11); in(7); in(8); out (); out ().

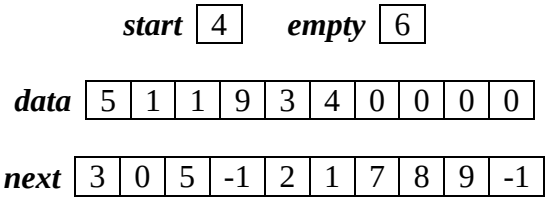
Recursion

8. (12 points) Consider the following code. What would the output be?

```
#include <iostream>
using namespace std;

int recursive(int x) {
  cout << x << ", ";
  if (x%2 == 0) {
    return recursive(x/2)*recursive(x/2);
  } else {
    return x;
  }
}

int gcd(int x, int y) {
  cout << "(" << x << "," << y << "), ";
  if (x == y) {
    return x;
  } else if (x < y) {
    return gcd(x,y-x);
  } else {
    return gcd(x-y,y);
  }
}

int main() {
  cout << "Part a:  ";
  int x = recursive(12);
  cout << x << endl;
  cout << "Part b:  ";
  int y = gcd(12,40);
  cout << y << endl;
}
```

Lists

9. (10 points) A singly-linked list of integers can be implemented using two arrays. The values of the two arrays at a particular index ***i*** describe one of the elements in the linked list, as follows: The first array (***data***) holds the integer stored at that element of the list. The second array (***next***) holds the index ***j*** where the next element of the list may be found. A separate variable ***start*** gives the index of the first element in the list, and a negative index indicates the end of the list. For example, the arrays shown below correspond to the list 3, 1, 4, 1, 5, 9:



Empty spaces in the array are also linked together in a stack-like structure so that they may be found easily. A variable called empty gives the index of the first empty space. When an element is inserted into the list, the first space from the empty list is used. When an element is deleted, its index is added to the front of the empty list.

a. Draw the state of the data structure described above if the number 7 is inserted into the list following the number 4.

b. Draw the state of the data structure if the number 5 is deleted from the original list (i.e., the list before part (a) above).

c. Comment on the advantages and disadvantages of this implementation of lists, compared with that used previously in class.