Midterm Topics

- → Stack
- → Queue
- ⇛ LLL, CLL, DLL
- ordered List
  - absolute (holes)
  - relative (No holes)
- Efficiency Discussion

CLOSED BOOK
CLOSED NOTES
1 hour 50min

Midterm *(100 points)*

CS 163: Data Structures

Name: ____;

Grade: ____________________

1. (25 points) Short Answer.Be brief! Limit your answers to 1-2 sentences:

a. When do we encounter the problem of *rightward drift*. Explain the circumstances.

b. List the advantages and disadvantages of using a dynamically allocated array to implement a ordered list ADT -- as a "relative" list with no holes:

Advantages:

Disadvantages.

c. When considering an array-based and a linked list implementation of a queue or a stack,

How does memory considerations influence your decision?

How does run-time efficiency influence your decision? Be precise.

2. (30 points) Write a function that will copy a doubly linked list (including the names in the list) to a new doubly linked list: Perform a complete copy of all data.

```
struct node {
  char * name;        //a doubly linked list of dynamic names
  node * next;
  node * prev;
};
```

(*) • copy a LLL

• Find the Largest item in a LLL

• Copy an Array into a new LLL

• copy a CLL into LLL

" " " into CLL

•

• Turn a CLL into a LLL

(*) • Remove the last node in a LLL

- with a tail |
- without a tail |

• Remove the last node in a CLL

(*) • Implement dequeue using a LLL

3. (30 points) Write the class header (just prototypes) for a queue of names (of varying length) implemented using a circular linked list

```
class queue {
     public:









     private:




};
```

Now, implement the destructor (Efficiency counts!)

What changes would need to be made if this were implemented using a dynamic array?

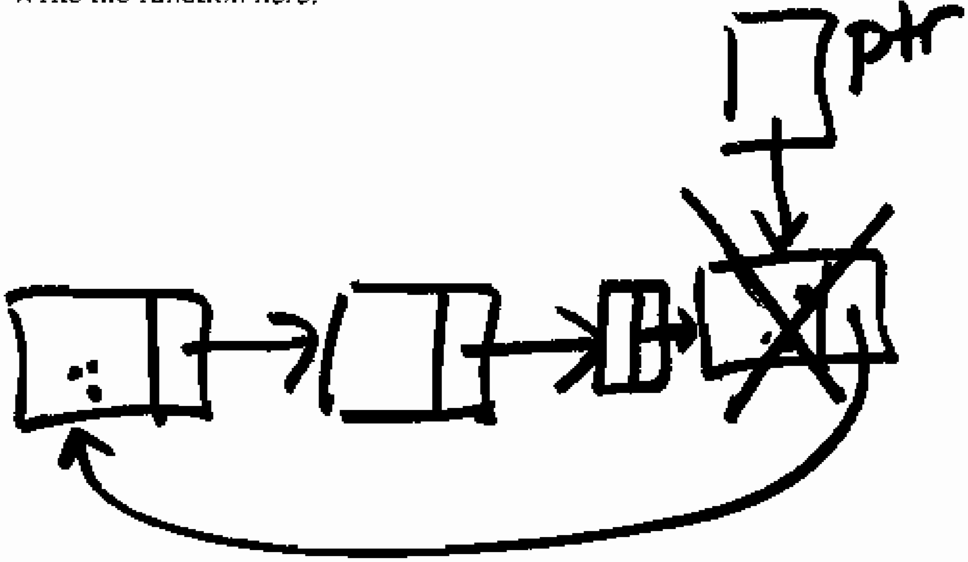
Describe the advantages and disadvantages (in 1-2 sentences) of using a circular linked list rather than a linear array to implement a queue   Discuss this in regards to efficiency --

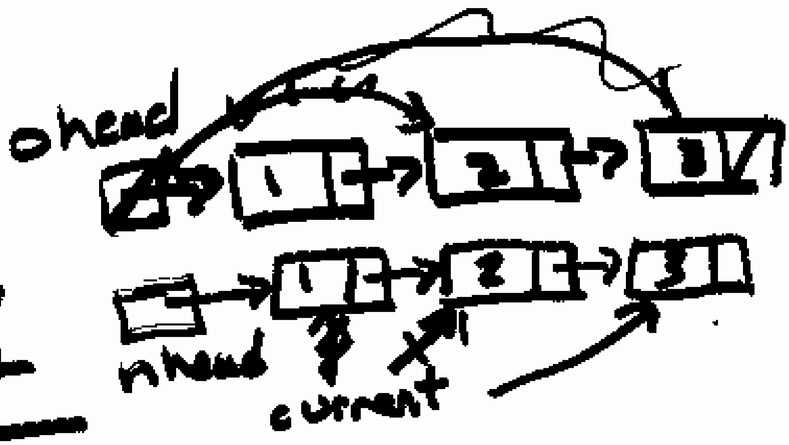
4. Dynamic Data Structures (15 points)

Assume you have a circular linked list. Let's assume it is a list of names (same as 2a). Write the code to remove the last node from this circular linked list (only given a head pointer as an argument):

Write the prototype here:____. _____

Write the function here:





hw #1

#1

```
int  Copy( node *&nhead,
           node *  ohead )
{  if ( !ohead)  // if ohead is NULL
   {     nhead = NULL;
         return 0;

   }
   nhead = new node;
   nhead->data = ohead->data;   ohead = ohead->next;
   node * current = nhead;
   while (   ohead   )
   {     current->next = new node;
         current = current->next;
         current->data = ohead->data;
         ohead = ohead->next;

   }
   current->next = NULL;
   return 1;
}
```

#2

LLL → Array

```
int copy ( int array[], node * ohead)
{
        if( !ohead)
             return ∅;

        ~~int index = ∅~~;
        ~~array~~
        for (int i=∅; ohead; ++i)
        {
             array[i] = ohead → data;
             ohead = ohead → next;

         }
         return 1;

    }
```

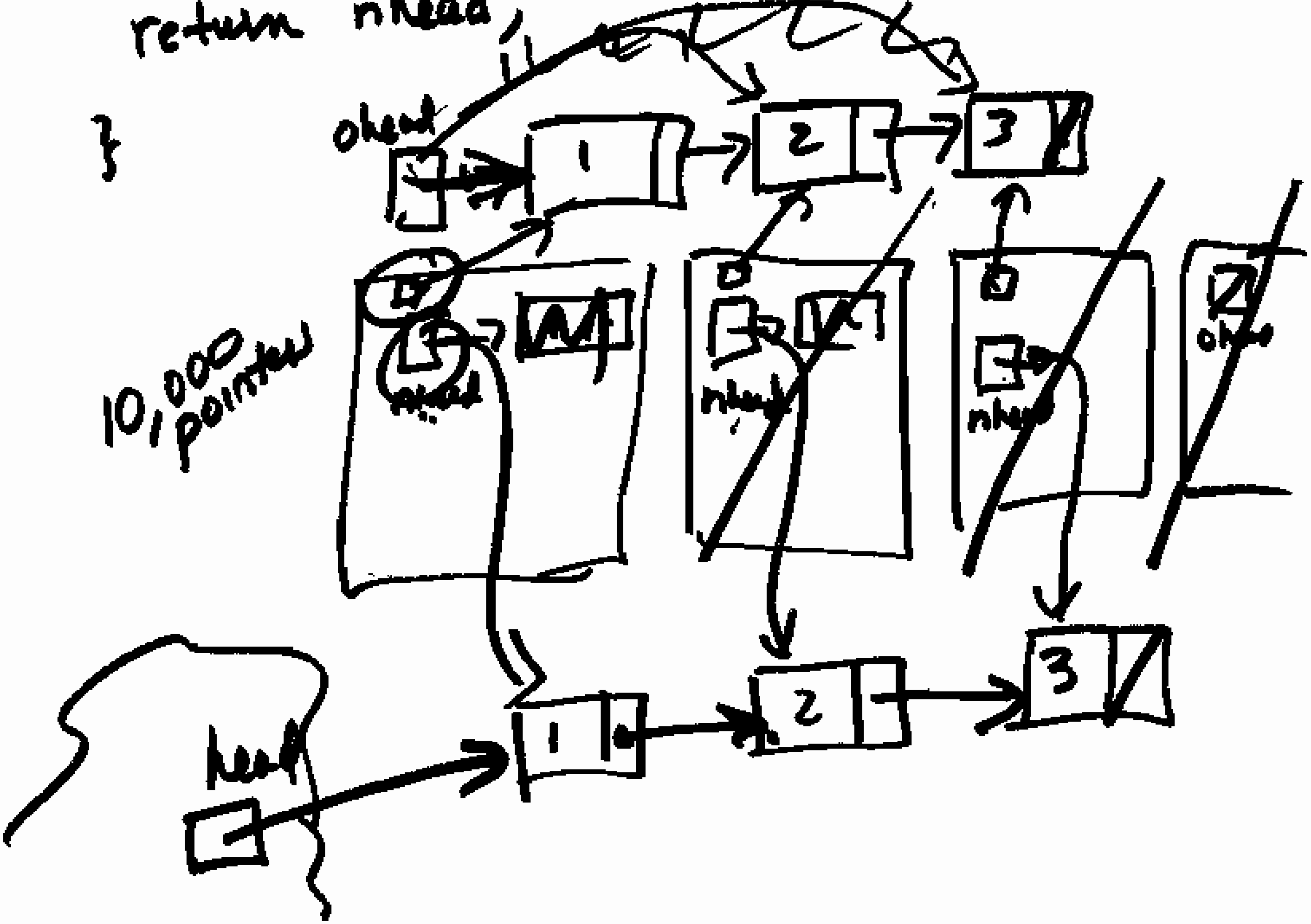
#3

LLL → LLL
using recursion !

```
int   copy (node * &nhead, node * ohead)
{     if ( !ohead )
      {      nhead = NULL;
             return Ø;

      }
      nhead = new node;
      nhead → data  = ohead → data;
   return 1 + copy (nhead → next, ohead → next);

 }
```

```
void  copy ( node * &nhead, node * ohead)
{ if ( !ohead)                          12 op + 8 f
       nhead = NULL                       ~ 20

 else { nhead = new node;
        nhead → data = ohead → data;
        copy ( nhead → next, ohead → next);
} }
```

```
node *   copy ( node * ohead)
{ if (ohead == NULL)  return NULL;
node *  nhead = new node;
        nhead → data = ohead → data;
                                        13 op + 8
                                          ~21

nhead→next = copy (ohead→next    );

return nhead;

}
```


ohead
1
2
3
10,000 pointer
nhead
nhead
nhead
ohead
1
2
3
head