CS 211 - 2/17/17

Friday, February 17, 2017 9:59 AM

Next week: Friday Lecture in LC - C4

Exam 1 in Lab on Wednesday 3/8

You are allowed to bring in any prewritten code, books, notes, and have access to the internet.

The code will require you to use one of the following 4 options:

- stack implemented as a dynamic array
- stack implemented as a linked list
- queue implemented as a dynamic array
- queue implemented as a linked list

Linked List & Recursion

IN the program, the path when found is stored in the stack from Finish to Start

To "flip the stack" using pure stack operations

Assume stack1 contains the "reverse" path

```
/* to "reverse" values on a stack */
Stack stack2 = init();

while ( !isEmpty (stack1) )
  {
    push ( stack2, top (stack1) );
    pop (stack1) ;
  }

/* to print out the value in stack2 */
while ( !isEmpty (stack2) )
  {
     print (  top (stack2) );
     pop (stack2);
```

```
  }
```

Solution 2: doable since we implement our own linked list
 - traverse the linked list to access the values without destroying/changing the stack.

/* to print the values from Finish to start */
Assuming the following stack/linked list definition

```
struct linkedStruct
{
 int elem;
 struct linkedStruct*  next;
};

typedef struct linkedStruct linked;
typedef linked* linkedPtr;
```

From <https://www3.cs.uic.edu/pub/CS211/LabsS17/lab5a.c>

```
linked* head;
…. /* code to build the "reverse" path */

printlist (head);

/* iterative list traversal */
void printlist ( linked* hd)
{
   while ( hd != NULL )
   {
      printf ( "%d, " , hd->elem);
      hd = hd->next;
   }
   printf ("\n");
}

 /* recursive list traversal */
void printlistR ( linked* nodePtr)
{
   if  ( nodePtr  != NULL )
   {
        printf ( "%d, " , nodePtr->elem );
        printListR ( nodePtr->next );
```

```
   }
   else
   {
       printf ("\n");
   }
}

 /* recursive reverse list traversal */
void printlistRR ( linked* nodePtr)
{
   if  ( nodePtr  != NULL )
   {       /* NOTE second R was missing during lecture */
           printListRR ( nodePtr->next );
           printf ( "%d, " , nodePtr->elem );
   }
}

/* recursive HELPER function */
void plRRHelper (linked* nodePtr)
{
  printlistRR (nodePtr);
  printf ("\n");
}

/* call from main () */
plRRHELPER ( head );
```