UCS 1302 Data Structures

Tutorial 2

--Dr. B. Prabavathy

1. Given a stack of integers, sort it in ascending order using another temporary stack.

input: [34, 3, 31, 98, 92, 23]

final sorted list: [3, 23, 31, 34, 92, 98]

Solution:

We follow this algorithm.

1. Create a temporary stack say **tmpStack**.
2. While input stack is NOT empty do this:
   * Pop an element from input stack call it **temp**
   * while temporary stack is NOT empty and top of temporary stack is greater than temp,   
     pop from temporary stack and push it to the input stack
   * push **temp** in temporary stack
3. The sorted numbers are in tmpStack

Element taken out: 23

input: [34, 3, 31, 98, 92]

tmpStack: [23]

Element taken out: 92

input: [34, 3, 31, 98]

tmpStack: [23, 92]

Element taken out: 98

input: [34, 3, 31]

tmpStack: [23, 92, 98]

Element taken out: 31

input: [34, 3, 98, 92]

tmpStack: [23, 31]

Element taken out: 92

input: [34, 3, 98]

tmpStack: [23, 31, 92]

Element taken out: 98

input: [34, 3]

tmpStack: [23, 31, 92, 98]

Element taken out: 3

input: [34, 98, 92, 31, 23]

tmpStack: [3]

Element taken out: 23

input: [34, 98, 92, 31]

tmpStack: [3, 23]

Element taken out: 31

input: [34, 98, 92]

tmpStack: [3, 23, 31]

Element taken out: 92

input: [34, 98]

tmpStack: [3, 23, 31, 92]

Element taken out: 98

input: [34]

tmpStack: [3, 23, 31, 92, 98]

Element taken out: 34

input: [98, 92]

tmpStack: [3, 23, 31, 34]

Element taken out: 92

input: [98]

tmpStack: [3, 23, 31, 34, 92]

Element taken out: 98

input: []

tmpStack: [3, 23, 31, 34, 92, 98]

final sorted list: [3, 23, 31, 34, 92, 98]

2. Create a data structure *twoStacks*that represents two stacks. Implementation of *twoStacks*should use only one array, i.e., both stacks should use the same array for storing elements. Following functions must be supported by *twoStacks*.  
push1(int x) –> pushes x to first stack   
push2(int x) –> pushes x to second stack  
pop1() –> pops an element from first stack and return the popped element   
pop2() –> pops an element from second stack and return the popped element  
Implementation of *twoStack*should be space efficient.

Solution:

Top1=-1

Top2 =size-1

if (top1 < top2 - 1), we can push an element, otherwise overflow for both the stacks

if(top1>=0) we can pop, otherwise underflow for stack 1

if(top2<size), we can pop, otherwise underflow for stack2

3. The Josephus problem is the following game:

N people, numbered 1 to N are sitting in a circle. Starting at person 1, a hot potato is passed. After M passes, the person holding the potato is eliminated, the circle closes and the game continues with the person who was sitting after the eliminated person picking up the hot potato. The last remaining person wins.

Write an application to find the winning position, given the values of N and M.

int main()

{

struct Queue\* queue = CreateQueue(100);

int players[6]={1, 2, 3, 4, 5, 6};

for (int i = 0; i < 6; i++) enqueue(queue, players[i]); printf("queue is\n");

for (int i=queue->front; i<=queue->rear; i++)

printf("%d ", queue->array[i]);

printf("\n");

int k=4;

**while (queue->size > 1)**

**{**

**for (int i = 1; i < k; i++)**

**enqueue(queue, dequeue(queue));**

**printf("\n%d is out",dequeue(queue));**

**}**

printf("\n\nwinner is %d",dequeue(queue));

return 0:

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*