# INTRODUCTION to ARTIFICIAL INTELLIGENCE 2015-2016 FALL SEMESTER LABORATORY MANUAL

## Experiment 3

**Data Structures (Linked List, Stacks, Queues) Implementations**

Stacks, and queues are elementary data structures that are based on pointers between some of the elements. These pointers are links to the first or last element in the set, to an element's previous or next element and so on.

### Stacks

- Stacks are dynamic sets in which the element removed from the set by DELETE operation is prespecified.
- It is an ordered group of homogeneous items.
- Items are added to and removed from the top of the stack

**LIFO property**: Last In, First Out

- The last item added would be the first to be removed.
- In stacks, the basic operations INSERT and DELETE are called PUSH and POP.
- A stack can be constructed using an array: array S[1..n]

attribute top[S] –indexes the element at the top

so a stack is an array S[1..top[S]].

### Queues

- In queues, the element that has been in the set for the longest time is deleted = first-in, first-out (FIFO).
- In queues, the INSERT and DELETE operations are called ENQUEUE and DEQUEUE. As the POP operation of the stack, the DEQUEUE operation does not take any parameters, it just deletes the element from the head of the queue.
- Stacks have only one pointer (the top[S] attribute that points to the top) but queues have two: head[Q] points to the head and tail[Q] points to the end of the queue.
- A queue can be realized efficiently using a circular array Q that ”wraps around”, i.e. so that when we are at the last index of an array the ”next” element is at index 1. The only thing we have to take care of is updating the head[Q] and tail[Q] attributes.
- When head[Q] = tail[Q], the queue is empty. Initially, head[Q] = tail[Q] = 1 When head[Q] = tail[Q] + 1, the queue is full.

**Exercise 1:**

Write a program to implement Stack data structure according to the following pseudocode given in Fig. 1.

```
STACK-EMPTY(S)
  if top[S] = 0
      return true
  else return false

PUSH(S, x)
  top[S] <- top[S] + 1
  S[top[S]] <- x

POP(S)
  if STACK-EMPTY(S)
      then error "underflow"
  else top[S] <- top[S] – 1
  return S[top[S] + 1]
```

**Fig. 1.** Pseudocode of Stack

**Exercise 2:**

Write a program to implement Queue data structure according to the following pseudocode given in Fig. 2.

```
ENQUEUE(Q, x)
  Q[tail[Q]] <- x
  if tail[Q] = length[Q]
      then tail[Q] <- 1
  else tail[Q] <- tail[Q] + 1

DEQUEUE(Q)
  x <- Q[head[Q]]
  if head[Q] = length[Q]
      then head[Q] <- 1
  else head[Q] <- head[Q] + 1
  return x
```

**Fig. 2.** Pseudocode of Queue