CSE 3010 – Data Structures & Algorithms

Lecture #13

Lecture #14

What will be covered today

- Implementation of stack data structure
    - Using arrays
    - Using linked list

Implementation of stack – Using arrays

```
// In stack.h
#define SIZE 100

typedef char ITEM;

typedef struct stack {
   ITEM stack[SIZE];
   int top;
} STACK;

STACK* create(STACK*);
int push(STACK*, ITEM);
ITEM pop(STACK*);
ITEM topOfStack(STACK*);
bool isEmpty(STACK*);
bool isFull(STACK*);
```

Important points to note when implementing stack using arrays

- Define the array and pointer to the top item as a structure
- Set top to -1 when an empty stack is created
- Increment top by 1 before pushing an item into the stack
- Decrement top by 1 after popping an item from the stack

Implementation of stack – Using linked list

```
// In stack_LL.h
#define SIZE 100
typedef char ITEM;
typedef struct snode {
   ITEM item;
   struct snode *next;
} SNODE;
typedef struct stack {
    SNODE *top_of_stack;
} STACK;
STACK* createStack();
SNODE* createNode(ITEM);
int push(STACK*, ITEM);
ITEM pop(STACK*);
ITEM top(STACK*);
bool isEmpty(STACK*);
bool isFull(STACK*);
```

Important points to note when implementing stack using linked list

- Create a node that has two parts – data and a pointer to a node of its type
- Create a stack that has a pointer to the node (top of the stack)
- Set top to NULL when an empty stack is created
- Set the top of stack to the last node item pushed into the stack
- Set the top of stack to the node after the top when an item is popped from the stack

[Refer class notes for detailed explanation on implementation of stack using arrays and linked list]