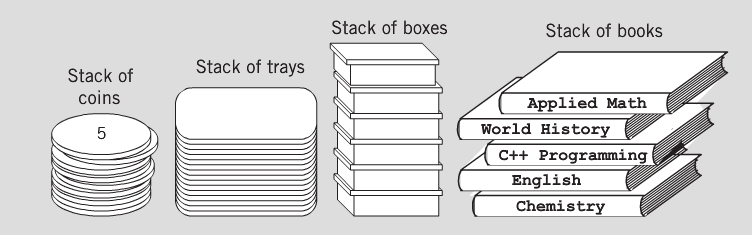
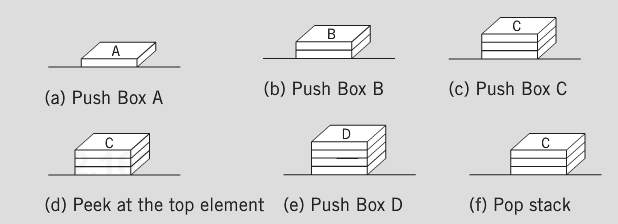
Stack

Push/Pop O(1)

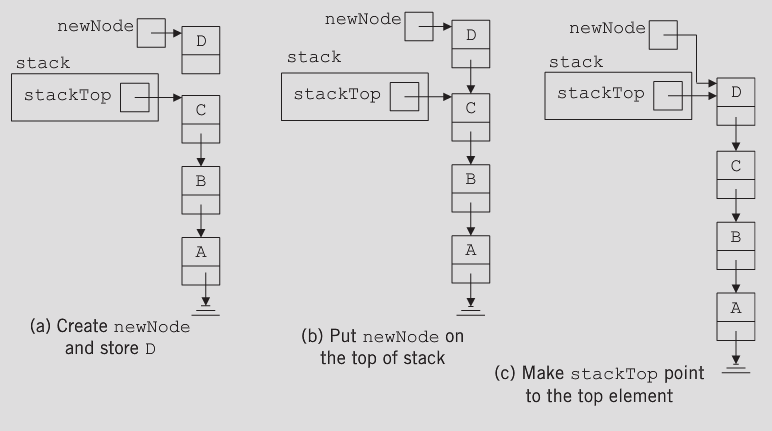
Search O(N)



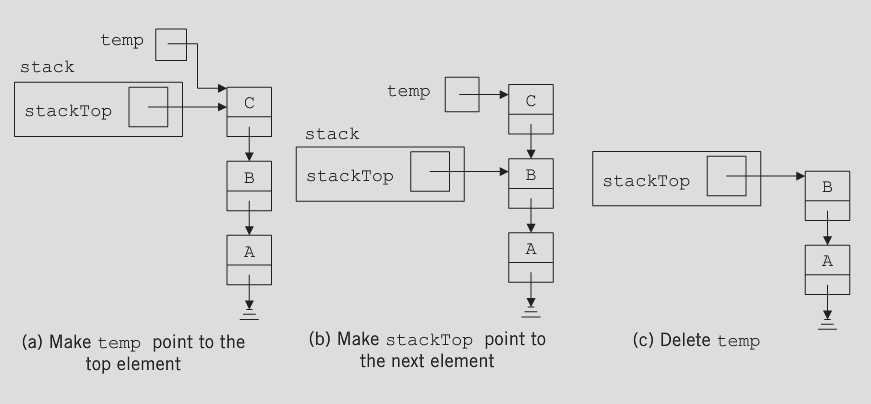


* stack – new nodes can be added and removed only at the top
  + similar to a pile of dishes
  + last-in/first-out (LIFO)
  + Bottom of stack indicated by a link member to **null**
  + constrained version of a linked list
* push
  + adds a new node to the top of the stack
* pop
  + removes a node from the top
  + stores the popped value
  + returns **true** if **pop** was successful

**Push operation:**



**Pop operation:**



struct node {

int item;

node \*next;

};

void pop(struct node \*&top){

struct node \*temp;

if(top){

temp=top;

top=top->next;

delete temp;

}

}

void push(int item, struct node \*&top){

struct node \*newNode=new node;

newNode->item=item;

newNode->next=top;

top=newNode;

}

int main(){

struct node \*top=NULL;

push(10,top);

pop(top);

}

***Other stack functions to consider***

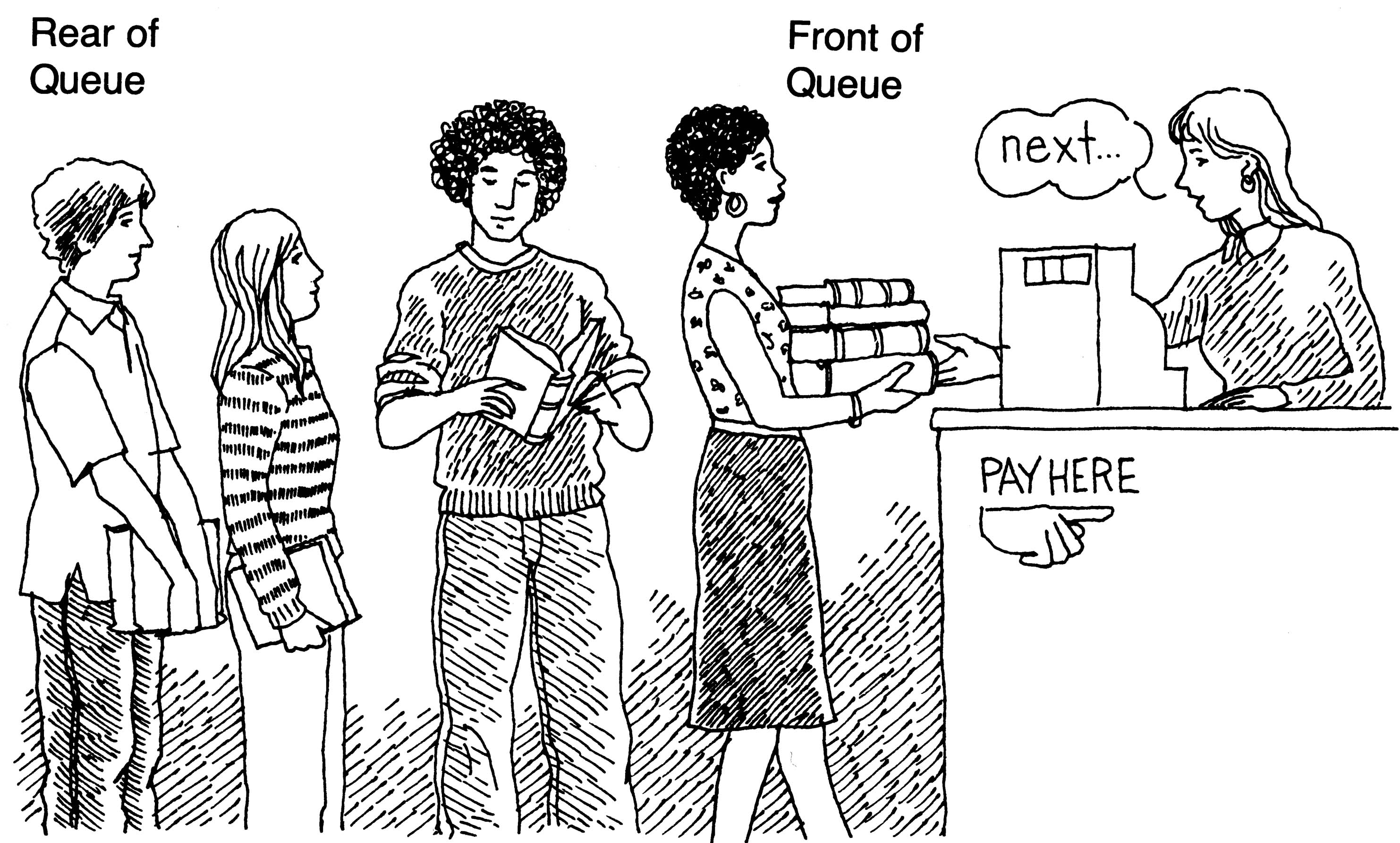
**InitializeStack**: Initializes the stack to an empty state

**isEmptyStack**: Determines whether the stack is empty. If the stack is empty, it returns the value true; otherwise, it returns the value false.

**isFullStack**: Determines whether the stack is full. If the stack is full, it returns the value true; otherwise, it returns the value false.

**Top**: Returns the top element of the stack. Prior to this operation, the stack must exist and must not be empty.

Queue

**

Queue: A data structure in which the elements are added at one end, called the rear, and deleted from the other end, called the front; a First In/First Out (FIFO) data structure.

A queue is a set of elements of the same type in which the elements are added at one end, called the back or rear, and deleted from the other end, called the front. For example, consider a line of customers in a bank, wherein the customers are waiting to withdraw/deposit money or to conduct some other business. Each new customer gets in the line at the rear. Whenever a teller is ready for a new customer, the customer at the front of the line is served. The rear of the queue is accessed whenever a new element is added to the queue, and the front of the queue is accessed whenever an element is deleted from the queue. As in a stack, the middle elements of the queue are inaccessible.

* queue – similar to a supermarket checkout line
  + *first-in, first-out (FIFO)*
  + nodes are removed only from the *front or head*
  + nodes are inserted only at the rear or *tail*
  + The insert and remove operations are known asenqueue and dequeue
  + Queue front examines the front element without deleting it
  + Queue rear examines the element at the rear of the queue without deleting it.
* Useful in computing
  + Print spooling, packets in networks, file server requests

**Insert O(1)**

**Search O(N)**

struct node {

int item;

node \*next;

};

void enqueue(int item, struct node \*&front, struct node\*&rear){

struct node \*newNode=new node;

newNode->item=item;

newNode->next=NULL;

if (rear){

rear ->next=newNode;

rear =newNode;

}else

front = rear =newNode;

}

void dequeue(struct node \*& front, struct node \*& rear){

struct node \*temp;

if(front){

temp= front;

front = front->next;

delete temp;

if (!front)

rear=NULL;

}

}

int main(){

struct node \* front, \* rear;

front = rear =NULL;

enqueue(5, front, rear);

dequeue(front, rear);

}

Quick summary

* *Linked lists* - insertions and removals made anywhere
* *Stacks* - insertions and removals made only at top of stack (LIFO)
* *Queues* - insertions made at the back and removals made from the front (FIFO)