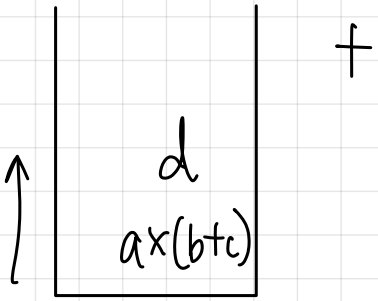
stack

ADT stack:

- isEmpty()
- push()
- pop() pop不传参数
- getTop()
- pop()

postfix Evaluator

abc+x d+



step in infix expression:

1. Append an operand to the end of an initially empty string postfixExpr

2. push (onto a stack

3. push an operator onto stack if stack is empty; otherwise pop operators and append them to postfixExpr as long as they have a precedence higher than or equal to >= that of the operator in the infix expression.

4. pop operators from stack and 運算

心得：在用 stack 做 prefix, infix, postfix 時理解了很久，一直看不太懂何時要 pop 何時要 push，最後多看了幾次影片後就有比懂，其中在表達式中有括號處理起來最容易。

Queue

佇列 = 排隊

ADT Queue

isEmpty

enqueue 新增

dequeue 移除

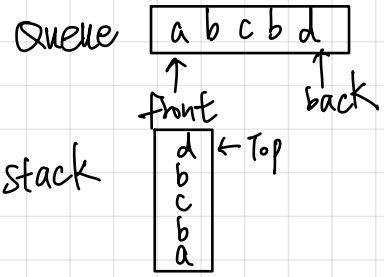
getFront 擷取

dequeue(out queueFront: QueueItemType) 擷取後移除

Recognize Palindromes 回文判斷

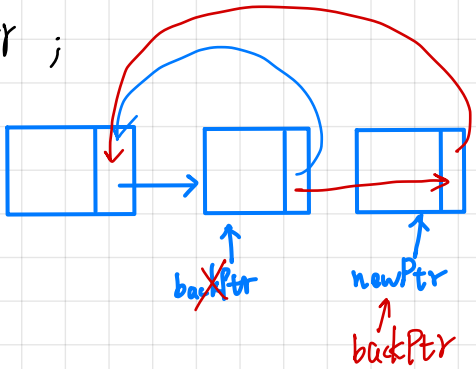
Using stack & queue

queue 的前端 vs. stack 的 Top

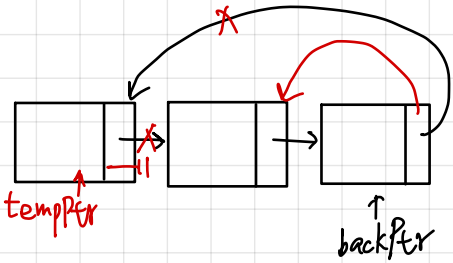



```
void Queue :: enqueue (const QueueItemType & newItem) {
    QueueNode * newPtr = new QueueNode;
    newPtr -> item = newItem;
    if(isEmpty()) newPtr -> next = newPtr;
    else {
```

```
    newPtr->next = backPtr->next;
    backPtr->next = newPtr;
  }//else
  backPtr = newPtr;
}// end enqueue
```




```
void Queue :: dequeue() throw (QueueException) {
  if (isEmpty())  throw...;
  else {
    QueueNode * tempPtr = backPtr->next;
    if(backPtr->next == backPtr)
        backPtr = NULL;
    else   backPtr->next = tempPtr->next;
    tempPtr->next = NULL;
    delete tempPtr;
  } //else
}// end dequeue.
```



心得：Queue 跟前面學過的 stack 滿像的，只是存資料跟取資料的順序有點不同，實作起來也不會很難。

排序：

stable sort vs. unstable sort

bubble　　selection

insertion　　quick

merge　　heap

radix　　$2\ 5_a\ 5_b\ 6\ 8_b\ 8_a\ 14\ 26\ 28\ 29$

順序可能被對調

selection Sort

單筆資料太大→搬動費時 才會選 Selection Sort

Mergesort　排序快，缺點需要額外的陣列存放排好的結果

Quick Sort　不是 stable

```
void radixsort(int A[], int first, int last){
  for(int base=1; (maxData/base)>0; base*=10){
    ...
    for(i=first; i<=last; i++)
      bucket[(A[i]/base)%10+1]++;
    // 算各餘數有幾個

    for(i=1; i<10; i++)
      bucket[i]+=bucket[i-1];
    for(i=first; i=last; i++)
      temp[bucket[(A[i]/base)%10]++]=A[i];

    for(i=first; i<=last; i++)
      A[i]=temp[i];
```

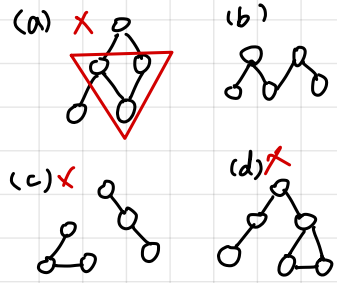
心得:

這單元教了很多排序法，也在作業中確實看到每個不同排序法效率上的差異，每個排序法都要想一下才可以理解，也在實作中看到 stable & unstable 分別是哪些

Tree

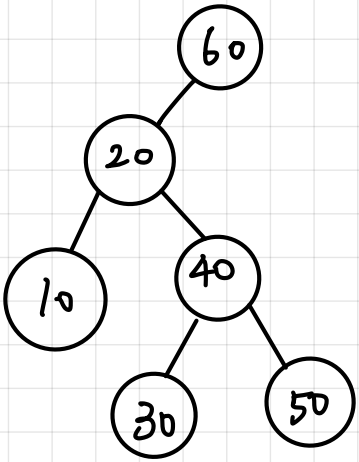
有大小階級關係可用 Tree.

不是 binary tree：



旋轉看是否是 binary tree

order：



preorder：60, 20, 10, 40, 30, 50, 70

inorder：10, 20, 30, 40, 50, 60, 70

postorder：10, 30, 50, 40, 20, 70, 60

要保持原樣 ⇒ 用 preoder 存檔案.

preorder + postorder 無法還原一顆唯一的 binary search tree

最高比較次數 ⇒ 樹高.

心得:

大一的計概有寫過 preorder, inorder, postorder 的 code
那時候理解了很久才寫出來, 第二次遇到
binary tree 我覺得理解起來有輕鬆一點