1. iterative :

15

```
int check (char A, int left, int right){
    while (left < right){
        if (A[left] != A[right])
            return -1; // not a palindrome
        left++;
        right--;
    }
    return 1; // palindrome
}
```

false 在 program 中 多用 "0"

recursive :

```
int check (char A, int left, int right){
    if (left >= right)
        return 1;
    if (A[left] != A[right])
        return -1;
    return check (A, left+1, right-1);
}
```

2.

9

```
void HanoiTower (char frompeg, char topeg, char auxpeg, int n)
{   if (n==0) /* the number of top disk*/ {
        output (frompeg, topeg, n); printf()
        return;
    }
    HanoiTower (frompeg, auxpeg, topeg, n-1);
    output (frompeg, topeg, n); printf()
    HanoiTower (auxpeg, topeg, frompeg, n-1);
}
```

10

```
3.  int findI(int A[], int x, int N){
        if( x < A[0]) return 0;
        if( x > A[N-1]) return N;
        left = 0;
        right = N;
        while(left <= right){
            mid = (left + right)/2;
        if(A[mid] > x){
            if(A[mid-1] < x)          ~~中止條件有誤吧~~
                return mid;
            right = mid-1;
        }else{
            if(A[mid+1] > x)
                return mid+1;
            left = mid+1;
        }
      }

4.  merge(A, B, C, m, n){
     p-q-r is the index of array A.B.C initialized as 0;
       while (p != m && q != n){
          switch(compare(A[p], B[q])){
           case '<': C[r]= A[p];
                     p++;
                     r++;
                     continue; 不適合放到 switch 裡面
           case '>':
           case '=': C[r]=B[q];
                     q++;
                     r++;
          }
       }attach the remaining part of A or B to C;  }
```

9

5.
$$\begin{array}{r} 5\,3\,6 \\ 4\,1\,5 \\ -\,3\,2\,3 \\ \hline 1\,(-1)\,2 \end{array}$$

$$\begin{array}{r} 3\,5\,6 \\ 1\,4\,5 \\ 2\,3\,3 \\ \hline (-1)\,1\,2 \end{array}$$

(a) $1069+18-6+2=1083$ ※ ✓

(b) $1069+(-1)\times5\times6+6+2=$ ~~1047~~ ※ 1095 -5

6. (a) 用箱子搬書，先放的後拿

搭电梯，先進後出 ✗

進火車車廂，先進後出

請使用 software or 演算法 相關例子.

-5 -3

(b) when add:

if (rear+1)%n == front) the queue is full. -1

when delete:

if (rear == front) the queue is empty. ✓

(C) if we use n elements, we can't judge whether the queue is empty or full while rear == front in both conditions. ✓

(d) $\log n < n < \log(n!) < n^2 < n^2\log n < \boxed{1.5^n} < n!$

→ $O(n\lg n)$.

(e) $(((a+b)-(c*d))+((e+f)*g))$

$\Rightarrow ab+cd*-ef+g*+$ ※ ✓

7.

tokens	6	2	/	3	−	4	2	*	+
stack	6	2 6	3	3 3	0	4 0	2 4 0	8 0	8

→ output 8 ✓

8. (a) $O(n\log n)$ ✓

(b) $O(\log n)$

-5