Assignment#5

1. Count the sum1 of all the nodes in the linked list, mark S1.

   Then get the sum2 from formula S2 = $\frac{n(n+1)}{2}$ (n stand for the nodes number).

   So, S1 – S2 should be the missing number.

2.

```
ListNode* Reverse (ListNode* pHead)
{
     ListNode* pReversedHead = NULL;

      ListNode* pNode = pHead;
      ListNode* pPrev = NULL;

    while(pNode != NULL)
     {
         ListNode* pNext = pNode->m_pNext;
         if(pNext == NULL)
             pReversedHead = pNode;

        pNode->m_pNext = pPrev;
        pPrev = pNode;
        pNode = pNext;
         }

        return pReversedHead;
    }
```

3. The one can put the address of arbitrary length strings on the queue.

And the time should be a constant number.

4. Can use binary search.

   int begin, end, middle;

   let begin = 1, end = n, middle = (begin + end) / 2;

do: if(middle > A[middle]) then first = middle + 1,

if(middle < A[middle]) then end = middle - 1;

otherwise, use the value of middle and return.

Until begin <= end.

If cannot find the I, then return -1.

The worst case computational complexity of this Algorithm should be O(log n).

5.

Set k as the position shifting begins.

Set n as the length of the array.

Set temp as an extra location.

First should

```
if(k <= size){place_array(k,array)}
else {k = k%n; place_array(k,array) }
int place_array(int k,int array[]){
    for(int j=0;j<k;j++){
        temp = array[0];
        for(int m=0; m < n-1; m++){
            array[m] = array[m+1];
        }
        array[n-1] = temp;
    }
```

```
    return 0;
}
```

6.

(1). Using Push() function push the first item of Q (Q.front()) to the empty stack S (Q.Dequeue()). And keep going until all items of Q into stack S.

(2). After step(1), the order of stack S: top is a_n, bottom is a_1. Then use S.pop() function and Q.Enqueue() function to let items in stack S move into queue Q.

(3). After step(2), the order of queue Q: front is a_n and tail is a_1. Then use push() function and Q.Dequeue() push all the items of queue Q into stack S.

So, after the step(3), the order of all the items has been preserved.

7. if rear > front : (rear –front + 1),

   if rear < front: (rear – front +1 + n)

   so: (rear – front + 1 + n) mod n

8.

```
    (a)
    int combinations(int m, int n)
    {
    if ((m == 0 || m==n)&&(m>=0))
        return 1;
```

```
else if (m > n || m<0 || n<0)
    return 0;
else
    return (combinations (m-1,n-1) + combinations (m,n-1));
}
```

(b)

```
int reverse(ListNode* p,ListNode* Head)
{
    if(p->next==NULL)
    {
        Head->next=p;
        return;
    }
    reverse (p->next,Head);
    p->next->next=p;
    p->next=NULL;
    return 0;
}
```

(c)

```
int reverseArray(int array[],int count) {
    if (count==1) return 0;
    int temp = array[0];
    for (int i = 0; i < count-1; i++) {
```

```
            array[i] = array[i+1];
        }
        array[count-1] = temp;
        reverseArray(array, count-1);
        return 0;
}
```

(d)

```
int bin_search(Object array[],int low, int high,int k)
{
    mid = (low+high) / 2;
    if(low>high)
        return -1;
    else{
        if(array [mid]==k)
            return mid;
        if(k> array [mid])
            return bin_search(array,mid+1,high,k);
        else
            return bin_search(array,low,mid-1,k);
    }
}
```

9. b) Stacks: Comes first, out last.