```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Node{
    int data;
    struct Node* next;
}Node;

Node* createNode(int data){
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

//=========第一题==========

//非递归处理
void deleteNode1(Node*head, int x){
    Node* current = head;
    Node* prev = NULL;
    while(current != NULL){
        if(current->data == x){
            if(prev == NULL){
                head = current->next;
            }else{
                prev->next = current->next;
            }
            free(current);
            return;
        }
        prev = current;
        current = current->next;
    }
}

//递归处理
Node* deleteNode2(Node* head, int x){
    if(head == NULL){
        return NULL;
    }
    if(head->data == x){
        Node* temp = head->next;
        free(head);
        return temp;
    }
    head->next = deleteNode2(head->next, x);
    return head;
}

//=========第二题==========
int findKth (Node* head, int k){
    Node* fast = head->next;
    Node* slow = head->next;

    for (int i = 0; i < k; i++){
        if (fast == NULL){
            return 0; //链表长度小于k
        }

        fast = fast->next;
    }

    while (fast != NULL){
        fast = fast->next;
        slow = slow->next;
    }

    printf("%d", slow->data);
    return 1;
}

//=========第三题==========
typedef struct QueueNode{
    int data;
    struct QueueNode* next;
}QueueNode;

typedef struct Queue{
    QueueNode* front;
    QueueNode* rear;
    int size;
}Queue;

void initQueue(Queue* q){
    q->front = NULL;
    q->rear = NULL;
    q->size = 0;
}

int isEmpty(Queue* q){
    return q->size == 0;
}

int getSize(Queue* q){
    return q->size;
}

void enqueue(Queue* q, int data){
    QueueNode* newNode = (QueueNode*)malloc(sizeof(QueueNode));
    newNode->data = data;
    newNode->next = NULL;
    if(isEmpty(q)){
        q->front = newNode;
        q->rear = newNode;
    }else{
        q->rear->next = newNode;
        q->rear = newNode;
    }
    q->size++;
}

int dequeue(Queue *q) {
    if (isEmpty(q)) {
        return -1;
    }
    QueueNode *temp = q->front;
    int data = temp->data;
    q->front = q->front->next;
    if (q->front == NULL) {
        q->rear = NULL;
    }
    free(temp);
    q->size--;
    return data;
}

int peekFront(Queue* q){
    if(isEmpty(q)){
        return -1;
    }
    return q->front->data;
}

int peekRear(Queue* q){
    if(isEmpty(q)){
        return -1;
    }
    return q->rear->data;
}

void clearQueue(Queue* q){
    while(!isEmpty(q)){
        dequeue(q);
    }
}

void freeQueue(Queue* q){
    clearQueue(q);
    free(q);
}

void printQueue(Queue* q){
    QueueNode* current = q->front;
    while(current != NULL){
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}
```