

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  typedef struct Node{
6      int data;
7      struct Node* next;
8  }Node;
9
10 Node* createNode(int data){
11     Node* newNode = (Node*)malloc(sizeof(Node));
12     newNode->data = data;
13     newNode->next = NULL;
14     return newNode;
15 }
16
17 //=====第一题=====
18
19 //非递归处理
20 void deleteNode1(Node*head, int x){
21     Node* current = head;
22     Node* prev = NULL;
23     while(current != NULL){
24         if(current->data == x){
25             if(prev == NULL){
26                 head = current->next;
27             }else{
28                 prev->next = current->next;
29             }
30             free(current);
31             return;
32         }
33         prev = current;
34         current = current->next;
35     }
36 }
37
38 //递归处理
39 Node* deleteNode2(Node* head, int x){
40     if(head == NULL){
41         return NULL;
42     }
43     if(head->data == x){
44         Node* temp = head->next;
45         free(head);
46         return temp;
47     }
48     head->next = deleteNode2(head->next, x);
49     return head;
50 }
51
52 //=====第二题=====
53 int findKth (Node* head, int k){
54     Node* fast = head->next;
55     Node* slow = head->next;
56
57     for (int i = 0; i < k; i++){
58         if (fast == NULL){
59             return 0; //链表长度小于k
60         }
61
62         fast = fast->next;
63     }
64
65     while (fast != NULL){
66         fast = fast->next;
67         slow = slow->next;
68     }
69
70     printf("%d", slow->data);
71     return 1;
72 }
73
74 //=====第三题=====
75 typedef struct QueueNode{
76     int data;
77     struct QueueNode* next;
78 }QueueNode;
79
80 typedef struct Queue{
81     QueueNode* front;
82     QueueNode* rear;
83     int size;
84 }Queue;
85
86 void initQueue(Queue* q){
87     q->front = NULL;
88     q->rear = NULL;
89     q->size = 0;
90 }
91
92 int isEmpty(Queue* q){
93     return q->size == 0;
94 }
95
96 int getSize(Queue* q){
97     return q->size;
98 }
99
100 void enqueue(Queue* q, int data){
101     QueueNode* newNode = (QueueNode*)malloc(sizeof(QueueNode));
102     newNode->data = data;
103     newNode->next = NULL;
104     if(isEmpty(q)){
105         q->front = newNode;
106         q->rear = newNode;
107     }else{
108         q->rear->next = newNode;
109         q->rear = newNode;
110     }
111     q->size++;
112 }
113
114 int dequeue(Queue *q) {
115     if (isEmpty(q)) {
116         return -1;
117     }
118     QueueNode *temp = q->front;
119     int data = temp->data;
120     q->front = q->front->next;
121     if (q->front == NULL) {
122         q->rear = NULL;
123     }
124     free(temp);
125     q->size--;
126     return data;
127 }
128
129 int peekFront(Queue* q){
130     if(isEmpty(q)){
131         return -1;
132     }
133     return q->front->data;
134 }
135
136 int peekRear(Queue* q){
137     if(isEmpty(q)){
138         return -1;
139     }
140     return q->rear->data;
141 }
142
143 void clearQueue(Queue* q){
144     while(!isEmpty(q)){
145         dequeue(q);
146     }
147 }
148
149 void freeQueue(Queue* q){
150     clearQueue(q);
151     free(q);
152 }
153
154 void printQueue(Queue* q){
155     QueueNode* current = q->front;
156     while(current != NULL){
157         printf("%d ", current->data);
158         current = current->next;
159     }
160     printf("\n");
161 }

```