```
#include <iostream>
#include <cassert>
using namespace std;

struct Node {
  int val;
  Node *next;
};

struct Queue {
  Node *frt;
  Node *last;
};

Queue *initQueue() {
  return new Queue {nullptr, nullptr};
}

bool isEmpty(Queue *q) {
  assert(q);
  return q->frt == nullptr;
}

void add(Queue *q, int val) {
  assert(q);
  Node *newnode = new Node {val, nullptr};
  if(q->frt == nullptr) {
    q->frt = newnode;
    q->last = newnode;
  } else {
    q->last->next = newnode;
    q->last = newnode;
  }
}

void remove(Queue *q) {
  assert(q);
  assert(q->frt);
  if(q->last == q->frt) {
    delete q->frt;
    q->last = nullptr;
    q->frt = nullptr;
```

```
  } else {
    Node *temp = q->frt->next;
    delete q->frt;
    q->frt = temp;
  }
}

bool check(Queue *q, int num) {
  assert(q);
  Node *cur = q->frt;
  while(cur) {
    if(cur->val == num) {
      return true;
    }
    cur = cur->next;
  }
  return false;
}

void nuke(Queue *q) {
  assert(q);
  while(q->frt) {
    remove(q);
  }
  delete q;
}

void print(Queue *q) {
  assert(q);
  Node *cur = q->frt;
  while(cur) {
    cout << cur->val << ' ';
    cur = cur->next;
  }
  cout << endl;
}
```

```
#include <iostream>
#include <cassert>
#include <vector>
using namespace std;

struct Queue {
  vector<int> vals;
};

Queue *initQueue() {
  return new Queue;
}

bool isEmpty(Queue *q) {
  assert(q);
  return q->vals.size() == 0;
}

void add(Queue *q, int val) {
  assert(q);
  q->vals.push_back(val);
}

void remove(Queue *q) {
  assert(q);
  //erase(q->vals.begin());
  int size = q->vals.size();
  for(int i = 0; i < size - 1; ++i) {
    q->vals[i] = q->vals[i + 1];
  }
  q->vals.pop_back();
}

bool check(Queue *q, int num) {
  assert(q);
  int size = q->vals.size();
  for(int i = 0; i < size; ++i) {
    if(q->vals[i] == num) {
      return true;
    }
  }
  return false;
```

```
}

void nuke(Queue *q) {
  assert(q);
  delete q;
}

void print(Queue *q) {
  assert(q);
  int size = q->vals.size();
  for(int i = 0; i < size; ++i) {
    cout << q->vals[i] << ' ';
    }
  cout << endl;
}
```