1.Cài đặt Queue

Queue dựa vào mảng :

#include <iostream>

using namespace std;

class Queue {

private:

  int front, rear, size;

  int\* array;

public:

  Queue(int size) {

    front = 0;

    rear = -1;

    this->size = size;

    array = new int[size];

  }

  void enqueue(int item) {

    if (rear == size - 1) {

      cout << "Queue is full\n";

      return;

    }

    rear++;

    array[rear] = item;

    cout << item << " added to queue\n";

  }

  void dequeue() {

    if (front > rear) {

      cout << "Queue is empty\n";

      return;

    }

    cout << array[front] << " removed from queue\n";

    front++;

  }

  void display() {

    if (front > rear) {

      cout << "Queue is empty\n";

      return;

    }

    cout << "Queue contains: ";

    for (int i = front; i <= rear; i++) {

      cout << array[i] << " ";

    }

    cout << endl;

  }

};

int main() {

  Queue q(5);

  q.enqueue(10);

  q.enqueue(20);

  q.enqueue(30);

  q.display();

  q.dequeue();

  q.display();

  q.enqueue(40);

  q.enqueue(50);

  q.enqueue(60);

  q.display();

  q.dequeue();

  q.dequeue();

  q.dequeue();

  q.dequeue();

  q.dequeue();

  q.dequeue();

  q.display();

  return 0;

}

Queue dựa vào danh sách liên kết :

#include <iostream>

using namespace std;

class Node {

public:

  int data;

  Node\* next;

};

class Queue {

private:

  Node\* front;

  Node\* rear;

public:

  Queue() {

    front = NULL;

    rear = NULL;

  }

  void enqueue(int item) {

    Node\* newNode = new Node();

    newNode->data = item;

    newNode->next = NULL;

    if (rear == NULL) {

      front = newNode;

      rear = newNode;

    } else {

      rear->next = newNode;

      rear = newNode;

    }

    cout << item << " added to queue\n";

  }

  void dequeue() {

    if (front == NULL) {

      cout << "Queue is empty\n";

      return;

    }

    Node\* temp = front;

    front = front->next;

    if (front == NULL) {

      rear = NULL;

    }

    cout << temp->data << " removed from queue\n";

    delete temp;

  }

  void display() {

    if (front == NULL) {

      cout << "Queue is empty\n";

      return;

    }

    cout << "Queue contains: ";

    Node\* current = front;

    while (current != NULL) {

      cout << current->data << " ";

      current = current->next;

    }

    cout << endl;

  }

};

int main() {

  Queue q;

  q.enqueue(10);

  q.enqueue(20);

  q.enqueue(30);

  q.display();

  q.dequeue();

  q.display();

  q.enqueue(40);

  q.enqueue(50);

  q.enqueue(60);

  q.display();

  q.dequeue();

  q.dequeue();

  q.dequeue();

  q.dequeue();

  q.dequeue();

  q.dequeue();

  q.display();

  return 0;

}

2.Bài DSA08006 Số 0 và số 9

Phân tích tính phù hợp :

Trong bài toán này, chúng ta cần tìm số nguyên dương nhỏ nhất X được tạo ra từ số 9 và 0 sao cho X chia hết cho số tự nhiên N đã cho. Để giải quyết bài toán này, chúng ta có thể sử dụng cấu trúc dữ liệu Queue.  
  
Ý tưởng của giải pháp này là xây dựng một Queue lưu trữ các số dương, bắt đầu từ số 1, sau đó ta lặp lại quá trình sau cho đến khi tìm được số X chia hết cho N:  
  
Lấy một số dương từ Queue (gọi là current\_number).  
Kiểm tra xem current\_number có chia hết cho N hay không. Nếu có, đó là kết quả cần tìm và kết thúc thuật toán.  
Nếu current\_number không chia hết cho N, ta thêm số 9 vào cuối của current\_number để tạo ra một số mới (gọi là new\_number1) và thêm số 0 vào cuối của current\_number để tạo ra một số mới khác (gọi là new\_number2).  
Đưa new\_number1 và new\_number2 vào cuối Queue.  
Ta sẽ tiếp tục quá trình lặp lại cho đến khi tìm được số X. Lưu ý rằng vì chúng ta đang tìm số nguyên dương nhỏ nhất X, nên khi tìm được số chia hết cho N, ta có thể dừng thuật toán ngay lập tức và trả về số đó.

Code :

#include <bits/stdc++.h>

using namespace std;

bool check(string s, int n) {

    int x = 0;

    for (int i = 0; i < s.length(); ++i) {

        x = (x \* 10 + (s[i] - '0')) % n;

    }

    return x == 0;

}

void testCase() {

    int n; cin >> n;

    queue<string> q;

    q.push("9");

    while (true) {

        string s = q.front();

        q.pop();

        if (check(s, n)) {

            cout << s;

            return;

        }

        q.push(s + "0");

        q.push(s + "9");

    }

}

int main() {

    ios\_base::sync\_with\_stdio(false);

    cin.tie(NULL); cout.tie(NULL);

    int T = 1; cin >> T;

    while (T--) {

        testCase();

        cout << "\n";

    }

    return 0;

}