# Aufgabe Floppy

| | |
|---|---|
| C header: | floppy_c.h |
| C++ header: | floppy.h |

Roxette als begeisterte Technologieenthusiastin, ist auf ein Array $v_0, v_1, v_2, \ldots, v_{N-1}$ aus $N$ **unterschiedlichen** Ganzzahlen gestossen. Da sie es sehr interessant findet, möchte sie das Array auf einer Diskette speichern. Allerdings musste Roxette wegen zu wenig freiem Speicherplatz einsehen, dass sie nicht das ganze Array auf der Diskette speichern kann. Stattdessen plant sie ein Array von Bits zu speichern, so dass sie Queries der folgenden Form beantworten kann:

query$(a, b) = id$, wobei $a \leq id \leq b$ und $v_{id} = \max(v_a, v_{a+1}, \ldots, v_{b-1}, v_b)$

In anderen Worten, ein Query gibt den Index des Elements mit dem höchsten Wert in einem gegebenen Subarray zurück.

Roxette braucht dafür deine Hilfe. Und zwar gleich doppelt. Zuerst gibt sie dir das interessante Array und du musst ihr sagen welche Bitfolge sie auf der Diskette speichern soll. Dann, wenn sie die Antwort auf gewisse Queries wissen will, gibt sie dir die Bitfolge, die du ihr gegeben hast um auf der Diskette zu speichern und die Queries welche sie beantwortet haben möchte. Du musst ihr dann die korrekte Antwort zu allen Queries geben.

## Interaktion

**Dies ist eine Aufgabe mit zwei Arten von Interaktionen!**
Es müssen zwei Funktionen implementiert werden. Die erste:

```
(C)   void read_array(int subtask_id, int N, int* v);
(C++) void read_array(int subtask_id, const std::vector<int> &v);
```

Diese Funktion wird, in der ersten Art von Interaktion, **genau einmal** aufgerufen und erhält das interessante Array von Roxette. In der Implementierung der Funktion muss folgende Funktion, welche vom Grader bereit gestellt wird, **genau einmal** aufgerufen werden.

```
(C)   void save_to_floppy(int L, char* bits);
(C++) void save_to_floppy(const std::string &bits);
```

Diese Funktion sagt Roxette welche Bitfolge sie auf ihrer Diskette speichern soll. Der Parameter `L` gibt an, wieviele Bits gespeichert werden sollen. Der Parameter `bits` muss ein String sein, der ausschliesslich die Zeichen `'0'` und `'1'` enthält.

Die zweite Funktion, die implementiert werden muss, ist folgende:

```
(C)   int* solve_queries(int subtask_id,
                         int N, int L, char* bits,
                         int M, int* a, int* b);
(C++) std::vector<int> solve_queries(int subtask_id,
                                     int N, const std::string &bits,
                                     const std::vector<int> &a,
                                     const std::vector<int> &b);
```

Diese Funktion wird in der zweiten Interaktion **genau einmal** aufgerufen und erhält die Länge des interessanten Arrays von Roxette, die Bitfolge, welche du Roxette vorher gegeben hast um sie auf der Diskette zu speichern und eine Liste von $M$ Queries.

Die Parameter für den `i`$^{\text{ten}}$ Query sind `a[i]` und `b[i]`.

Diese Funktion muss ein Array von $M$ Ganzzahlen zurückgeben, welche die Antwort auf die Queries sind. Die $\texttt{i}^{\text{te}}$ Ganzzahl muss die Anwort auf den $\texttt{i}^{\text{ten}}$ Query sein.

**Wichtig:** Das Programm wird zweimal laufen gelassen, je einmal für jede Interaktion. Somit sind alle Daten, die bei der Ausführung der ersten Interaktion berechnet wurden, nicht zugänglich in der zweiten Ausführung.

## Limits

- $-10^9 \leq v_i \leq 10^9$ für alle $0 \leq i \leq N-1$
- $1 \leq L \leq 200\,000$

## Teilaufgabe 1 (7 Punkte)

- $1 \leq N \leq 500$
- $0 \leq v_i < N$ für alle $0 \leq i \leq N-1$
- $1 \leq M \leq 1\,000$
- Die Punkte für diese Teilaufgabe sind 7 falls alle Tests korrekt gelöst sind, ansonsten gibt es 0 Punkte.

## Teilaufgabe 2 (21 Punkte)

- $1 \leq N \leq 10\,000$
- $1 \leq M \leq 20\,000$
- Die Punkte für einen der Tests sind $\min(1, \frac{1}{2^{\frac{L}{N}-1-\log_2 N}})$. Die Punktzahl für die Teilaufgabe ist das Minimum der Punkte aller Tests, mit 21 multipliziert.

## Teilaufgabe 3 (72 Punkte)

- $1 \leq N \leq 40\,000$
- $1 \leq M \leq 80\,000$
- Die Punkte für einen der Tests sind $\min(1, \frac{1}{2^{\frac{L}{N}-2}})$. Die Punktzahl für die Teilaufgabe ist das Minimum der Punkte aller Tests, mit 72 multipliziert.

## Beispielinteraktion

In der ersten Interaktion wird die erste Funktion aufgerufen:

```
read_array(
  /* subtask_id = */ 3,
  /* v          = */ {40, 20, 30, 10});
```

Diese Funktion wiederrum kann die folgende Bitfolge wählen, um sie auf der Diskette zu speichern:

```
save_to_floppy(
  /* bits = */ "001100");
```

Die Ausführung des Programms wird nun terminiert und alle Daten, die im Speicher abgelegt sind, werden gelöscht.

In der zweiten Interaktion wird die zweite Funktion aufgerufen:

```
solve_queries(
  /* subtask_id = */ 3,
  /* N          = */ 4,
  /* bits       = */ "001100",
  /* a          = */ {0, 0, 0, 0, 1, 1, 1, 2, 2, 3},
  /* b          = */ {0, 1, 2, 3, 1, 2, 3, 2, 3, 3});
```

Diese Funktion sollte ein Array von $M$ Ganzzahlen zurückgeben:

```
  {0, 0, 0, 0, 1, 2, 2, 2, 2, 3}
```