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

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

struct Seq {
  Node *first;
  Node *last;
  int size;
};

void insert(Seq &seq, string str, int pos) {
  assert(pos < seq.size);
  Node *newnode = new Node {str, nullptr};
  Node *prev = nullptr;
  Node *cur = seq.first;
  for(int i = 0; i < pos; ++i) {
    prev = cur;
    cur = cur->next;
  }
  if(!prev) {
    newnode->next = seq.first;
    seq.first = newnode;
  } else {
    newnode->next = prev->next;
    prev->next = newnode;
  }
  seq.size++;
}

void insert(vector<string> &seq, string str, int pos) {
  assert(pos < seq.size());
  seq.push_back("");
  for(int i = seq.size() - 1; i >= pos; --i) {
    seq[i] = seq[i - 1];
  }
```

```
  seq[pos] = str;
}

void append(Seq &seq, string str) {
  Node *newnode = new Node {str, nullptr};
  if(seq.first) {
    seq.last->next = newnode;
    seq.last = newnode;
  } else {
    seq.first = newnode;
    seq.last = newnode;
  }
}

void append(vector<string> &seq, string str) {
  seq.push_back(str);
}

string at(Seq &seq, int pos) {
  assert(pos < seq.size);
  Node *cur = seq.first;
  while(pos > 0) {
    cur = cur->next;
    pos--;
  }
  return cur->val;
}

string at(vector<string> &seq, int pos) {
  assert(pos < seq.size());
  return seq[pos];
}
```

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

```
struct Node {
  int num;
  string name;
  bool deleted;
  Node() : num{0}, name{""}, deleted{true} {}
};

int elemt = 0;

void initHt(vector<Node> &table, int size) {
  table.resize(size);
}

int f(int num, int size) {
  return num % size;
}

void insert(vector<Node> &table, string name, int num) {
  int pos = f(num, table.size());
  assert(table.size() > elemt);
  while(!table[pos].deleted) {
    pos = (pos + 1) % size;
  }
  table[pos].num = num;
  table[pos].name = name;
  table[pos].deleted = false;
  elemt++;
}
```