.vimrc

```
set nu ai sta sw=2 ts=2 hls is

filetype plugin indent on
syn on

inoremap {<cr> {<cr>}<esc>O
nnoremap <space>` :b#<cr>
nnoremap ; :
```

.bashrc

```
set -o vi
alias y='setxkbmap -option caps:swapescape'
alias x='setxkbmap -option'
alias v='vim'

c() {
    g++ -std=c++20 -Wall -Wextra -Wshadow -Wno-sign-
conversion -Wconversion -Wfloat-equal -
fsanitize=undefined,address -ggdb3 -fno-sanitize-recover -
DLOCAL -DDEBUG -D_GLIBCXX_DEBUG $1.cpp -o $1
}
```

sol.cpp

```
#include <bits/stdc++.h>

using namespace std;

#ifdef LOCAL
#include "debug.h"
#else
#define debug(...) {}
#endif

#define x first
#define y second
#define ir(a, x, b) ((a) <= (x) && (x) <= (b))
#define vec vector
#define rep(i, a, b) for (int i = a; i < (b); ++i)
#define all(x) (x).begin(), (x).end()

using ll = long long;

int main() {
    cin.tie(0)->sync_with_stdio(0);

    return 0;
}
```

debug.h

```
ostream& operator<<(ostream& os, const pair<auto, auto>&
x);

template<typename T, typename = T::value_type>
ostream& operator<<(ostream& os, const T& x) requires (!
same_as<T, string>) {
  os << '{';
  string sep;
  for (const auto& v : x) {
    os << sep << v;
    sep = ", ";
  }
  return os << '}';
}

ostream& operator<<(ostream& os, const pair<auto, auto>& x)
{
  return os << '(' << x.first << ", " << x.second << ')';
}

void __print(const auto&... x) {
  ((cerr << ' ' << x), ...);
  cerr << endl;
}

#define debug(...) cerr << '[' << #__VA_ARGS__ << "]:";
__print(__VA_ARGS__)
```