ad-1. 連結リスト

（C 言語によるアルゴリズムとデータ構造）（全６回）

URL: https://www.kkaneko.jp/pro/ad/index.html

金子邦彦

1-1. 連結リスト

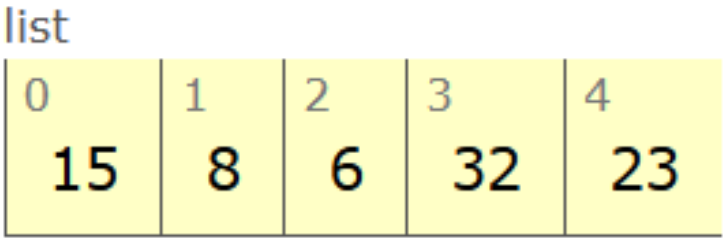
リストの性質

- **順序のあるデータ**
- 要素の**削除**，**挿入**により**サイズが増減**する

list

0	1	2	3	4
15	8	6	32	23

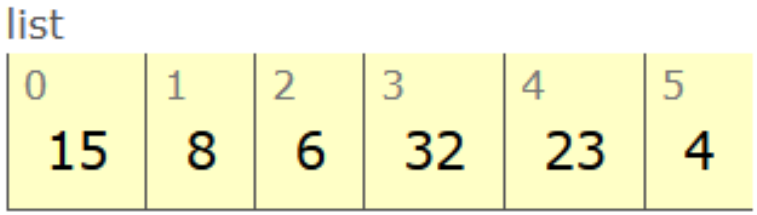
リストの組み立て，要素の挿入，要素の削除



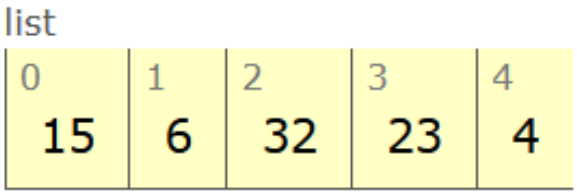
リストの組み立て

```
x = [15, 8, 6, 32, 23];
```

要素の挿入
４を挿入



要素の削除
８を削除



リストの種類

処理	配列によるリスト	連結リスト
要素の末尾への追加	**高速**	**高速**
要素の末尾以外への追加	低速	**高速**
要素の末尾からの削除	**高速**	**高速**
要素の末尾以外からの追加	低速	**高速**
添え字による要素アクセス	**高速**	低速

・「高速」，「低速」は，配列によるリスト，連結リストを比べたとき，どちらが相対的に早いかの傾向を示すもの
・連結リストでの要素の追加，削除は，追加，削除すべきレコードのアドレスが分かっているものとする

- 配列によるリスト

  あらかじめ，ある程度の長さの配列を作り，
  その中にリストを格納するもの

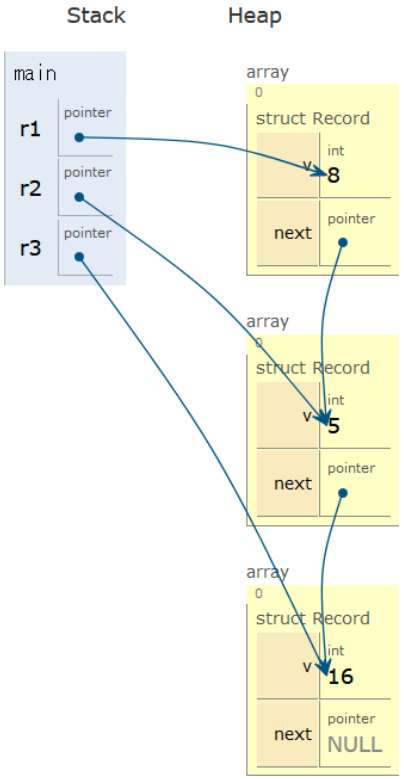
- 連結リスト

  データを格納するためのメモリを挿入のたびに確保．そして，削除のたびに解放.

連結リストとは

レコードを次の２つで構成

- 要素を格納する**セル**
- **リスト**中の次の**レコード**を指す**ポインタ**を格納する**セル**



1-2.「リスト」を実習できるオンラインサイトの紹介

「リスト」を実習できる
オンラインサイトの紹介

リストとは，順序の付いたデータの並び

パソコン演習

① Chrome ウェブブラウザを起動する

② 次の URL を開く

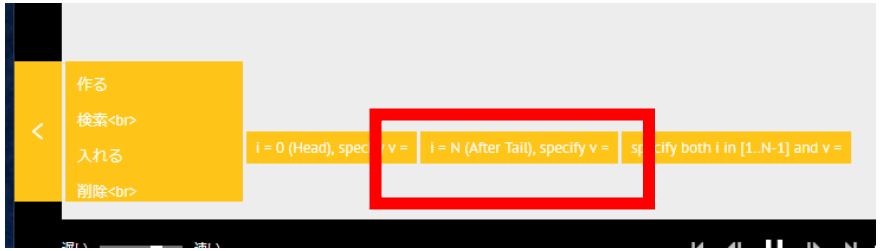
https://visualgo.net/ja

③ 「**連結リスト**」をクリック

パソコン演習

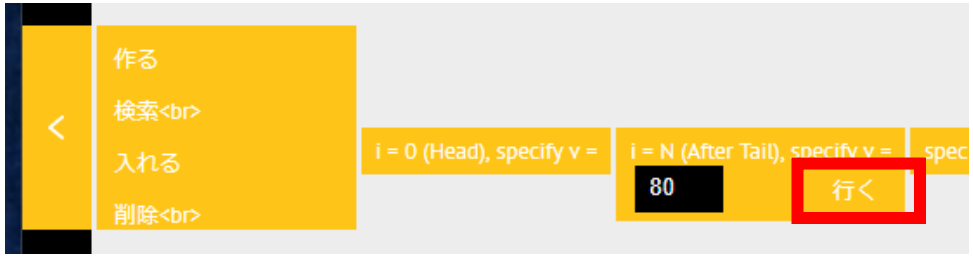
④ 説明が出る．**ESC キー**を押して，説明を消す

⑤　左下のメニューで「**入れる**」をクリックし，
「**i = N (After tail), specify v =**」を選ぶ

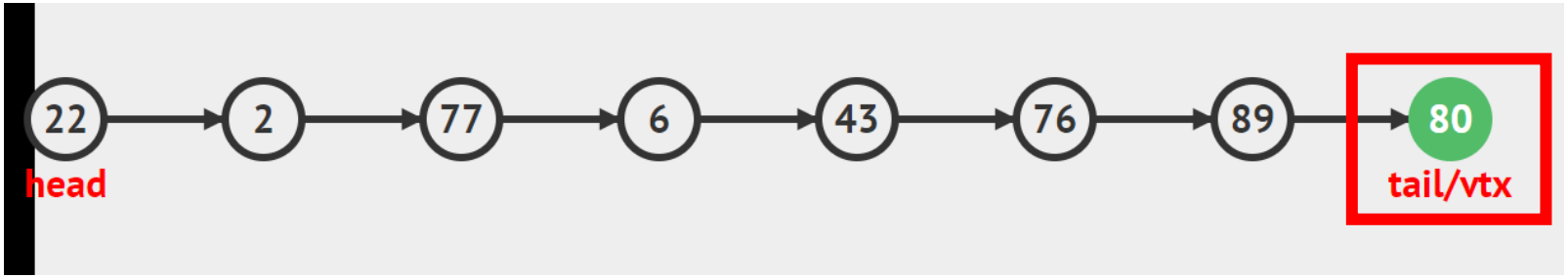


パソコン演習

⑥　値が「80」のように表示されるので，確認したら「**行く**」をクリック



⑦ **<u>末尾にデータが増える</u>**ので，確認する



1-3. 実習

実習の指示

- 資料：**１５～２４**
- C Tutor に関する次のことを理解しマスターする
  - C Tutor の起動手順
  - C Tutor の画面構成
  - C Tutor は，オンラインのプログラム開発環境であること

実習

① **ウェブブラウザ**を起動する

② **C Tutor** を使いたいので，次の URL を開く

http://www.pythontutor.com/

※ **Internet Explorer でうまく動かない**場合がある

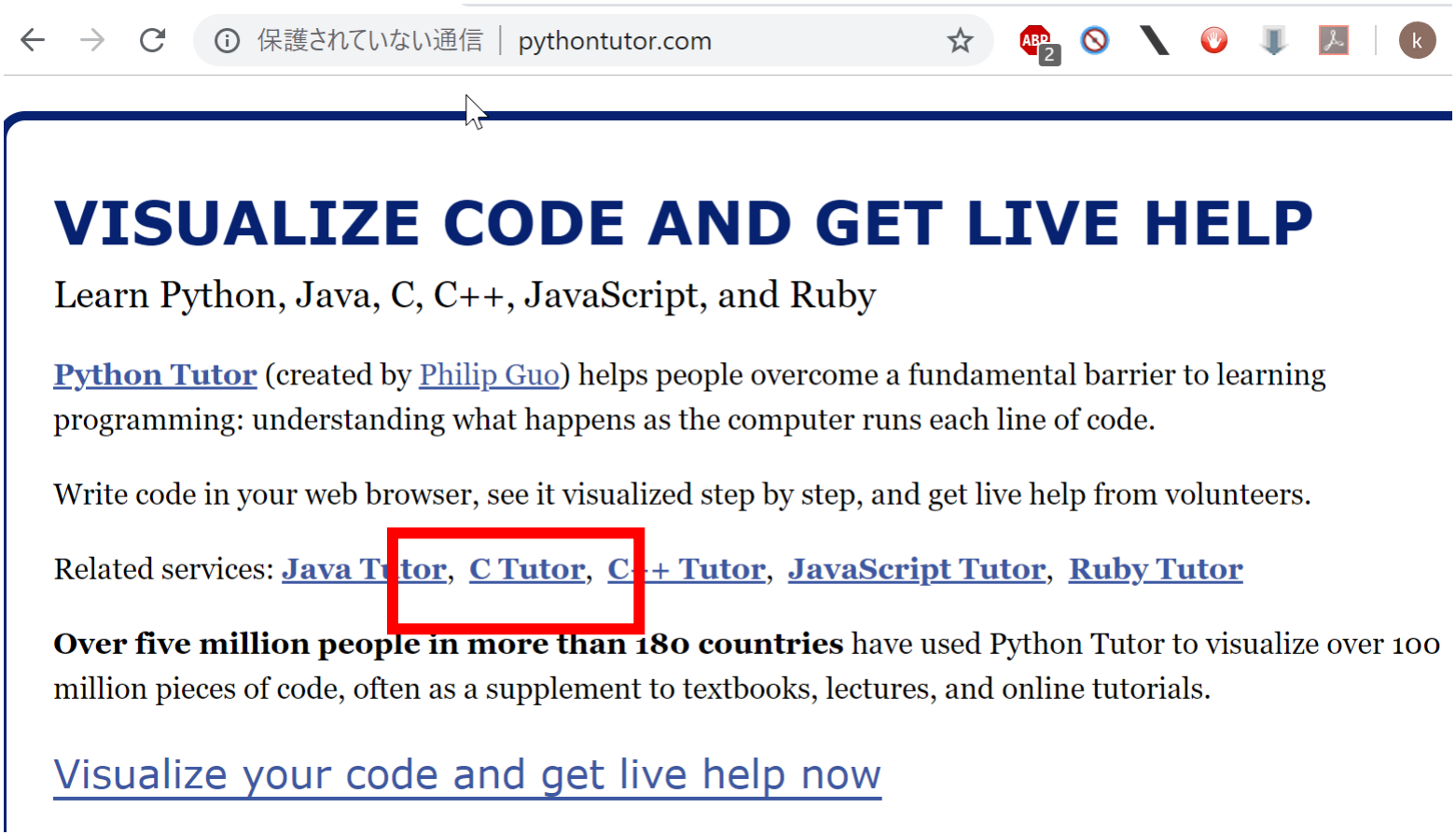
→ うまく動かないときは **Google Chrome** を試してください

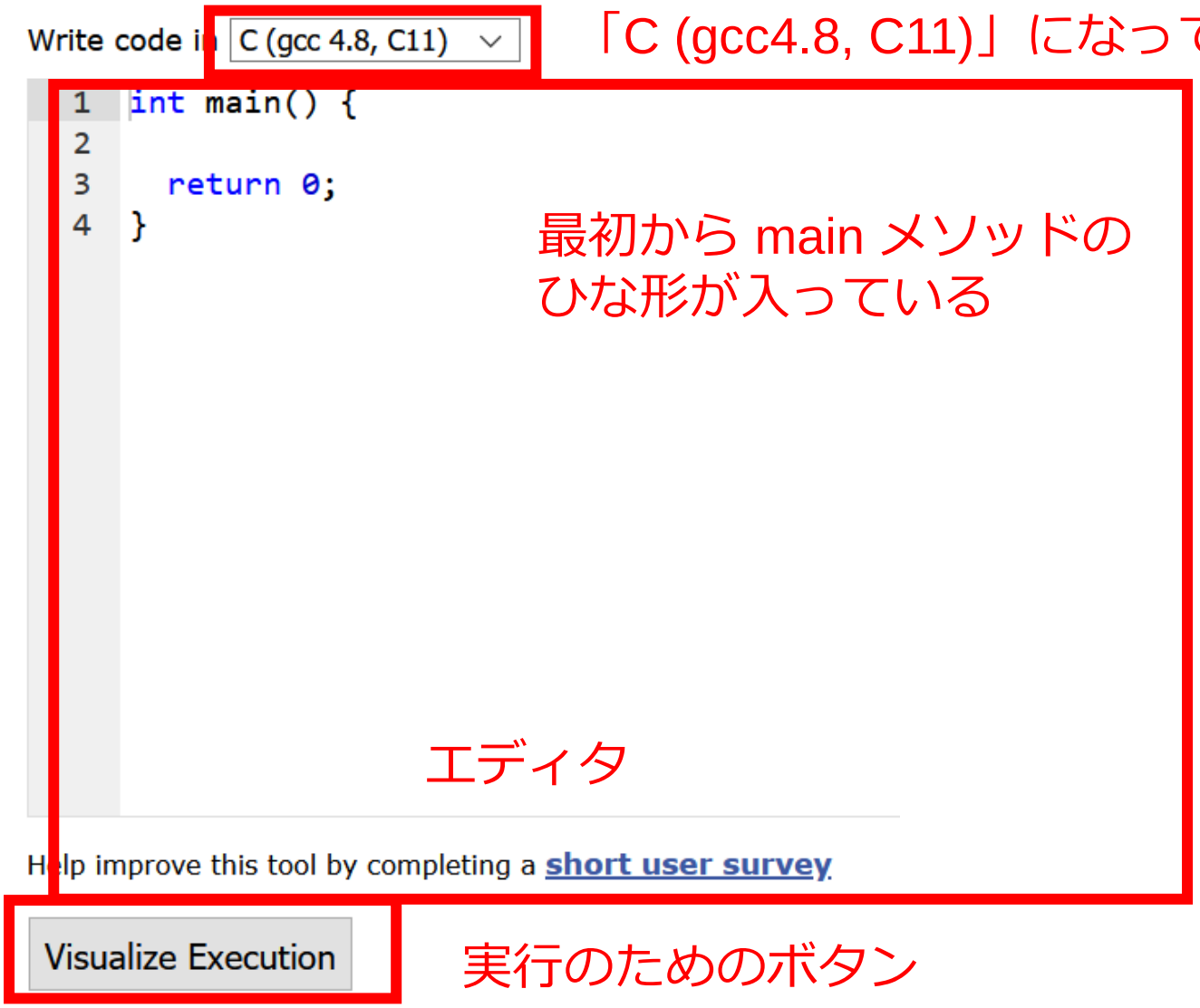
※ 途中で**「Server Busy・・・」というメッセージが出る**ことがある.

→ 混雑している．**少し（数秒から数十秒）待つ**と自動で表示が変わる（変わらない場合には，操作をもう一度行ってみる）

※ 日本語モードはない．英語で使う

③ 「C Tutor」をクリック




Write code in
C (gcc 4.8, C11)
「C (gcc4.8, C11)」になっている
1 int main() {
2
3 return 0;
4 }
最初から main メソッドの
ひな形が入っている
エディタ
Help improve this tool by completing a short user survey
Visualize Execution
実行のためのボタン

実習の指示

- 資料：**１９～２４**
- 次のことを理解しマスターする
  - 連結リストの作成
  - 要素の末尾への追加
  - 要素の末尾からの削除

連結リストの作成

① 次のプログラムを使う

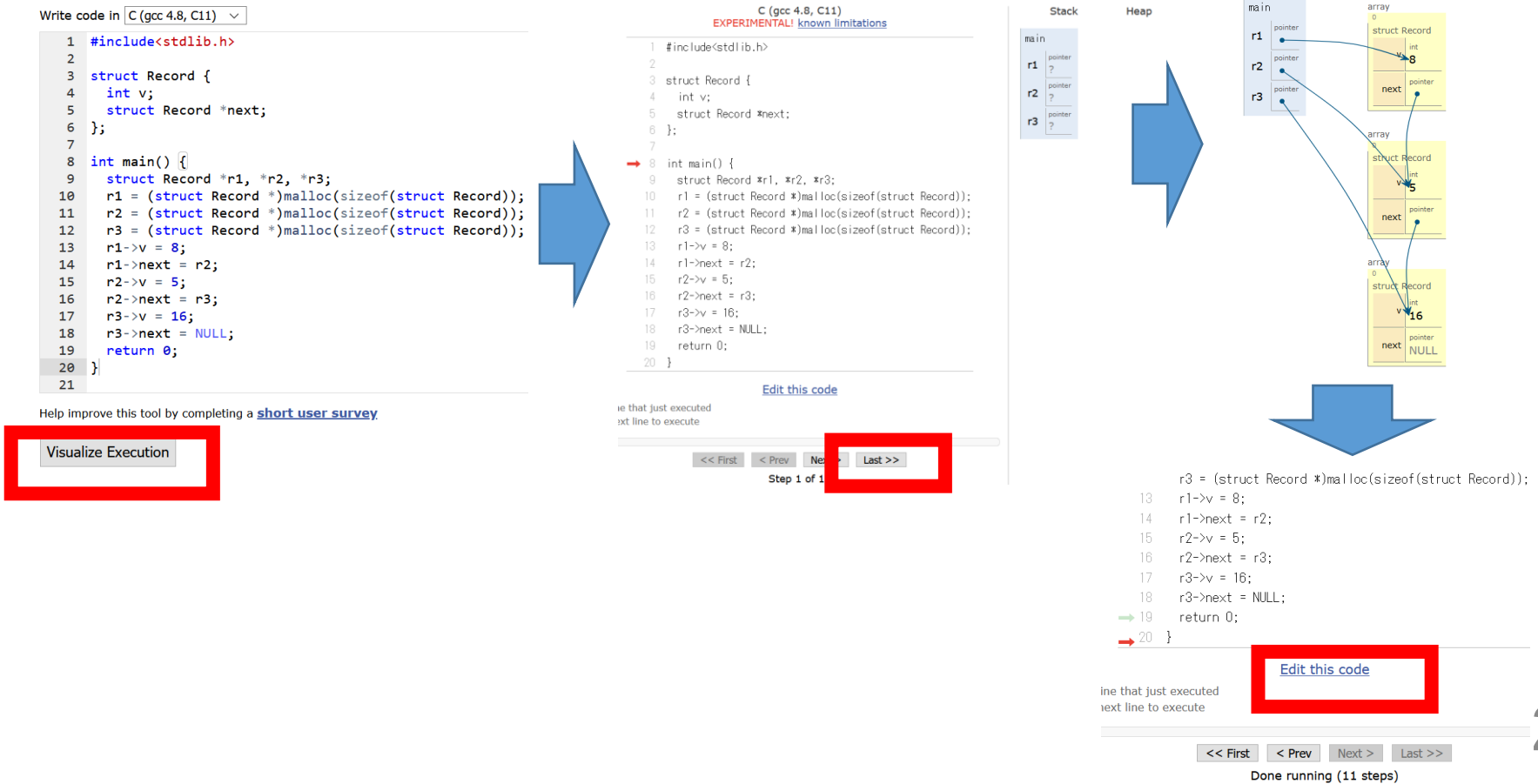
```c
#include<stdlib.h>

struct Record {
  int v;
  struct Record *next;
};

int main() {
  struct Record *r1, *r2, *r3;
  r1 = (struct Record *)malloc(sizeof(struct Record));
  r2 = (struct Record *)malloc(sizeof(struct Record));
  r3 = (struct Record *)malloc(sizeof(struct Record));
  r1->v = 8;
  r1->next = r2;
  r2->v = 5;
  r2->next = r3;
  r3->v = 16;
  r3->next = NULL;
  return 0;
}
```

②「Visualize Execution」をクリック.

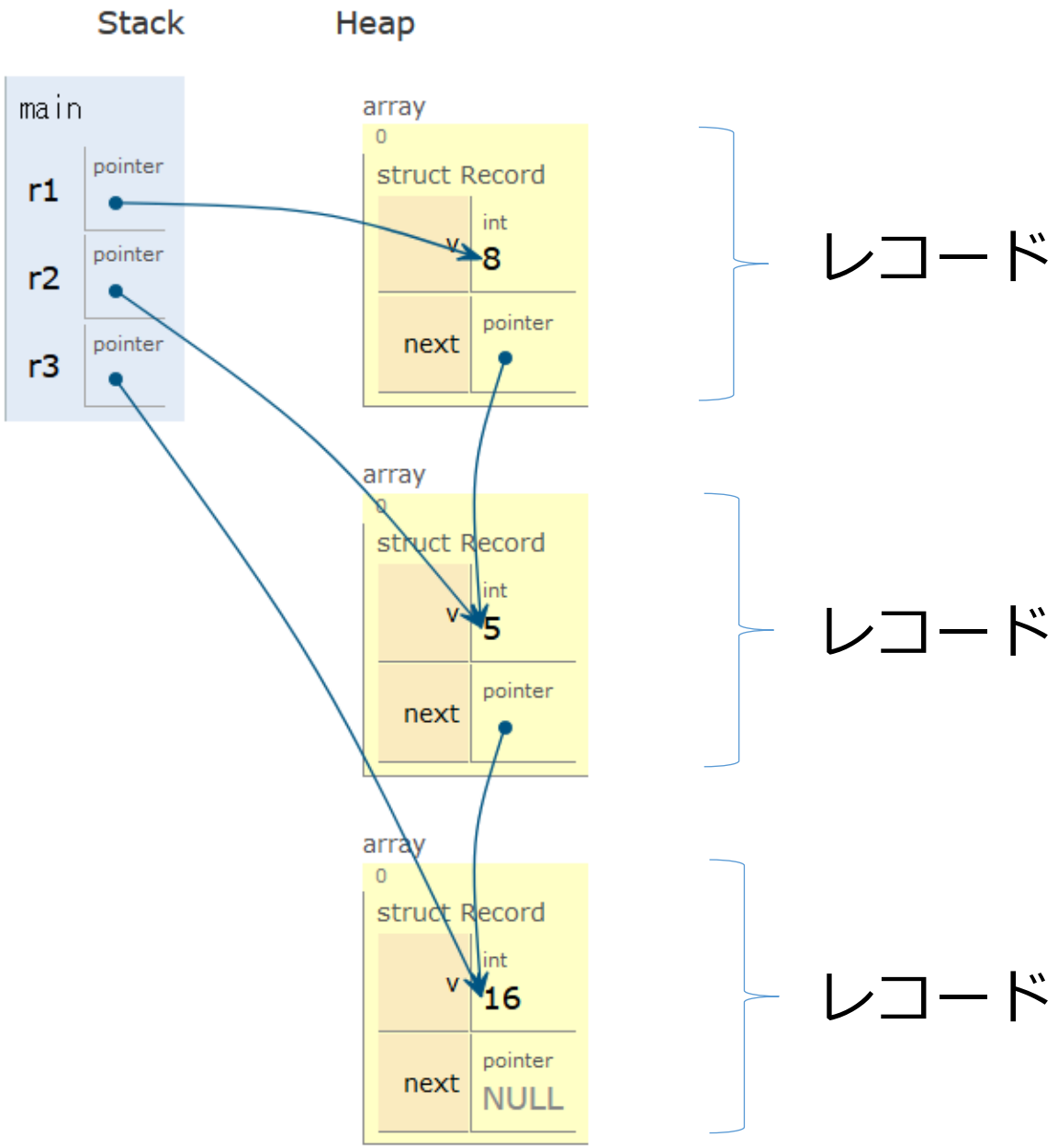
「Last」をクリック.

結果を確認する.

「Edit this code」をクリックして戻る

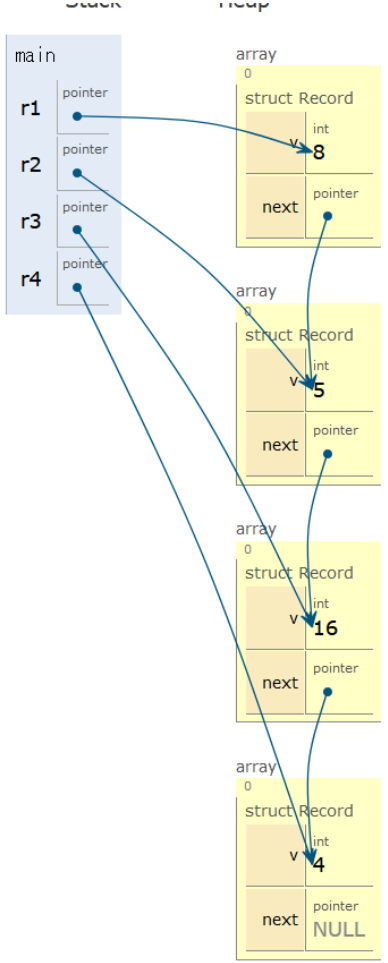


実行結果


Stack
Heap
main
r1
pointer
r2
pointer
r3
pointer
array
0
struct Record
int
v
8
next
pointer
array
0
struct Record
int
v
5
next
pointer
array
0
struct Record
int
v
16
next
pointer
NULL
レコード
レコード
レコード

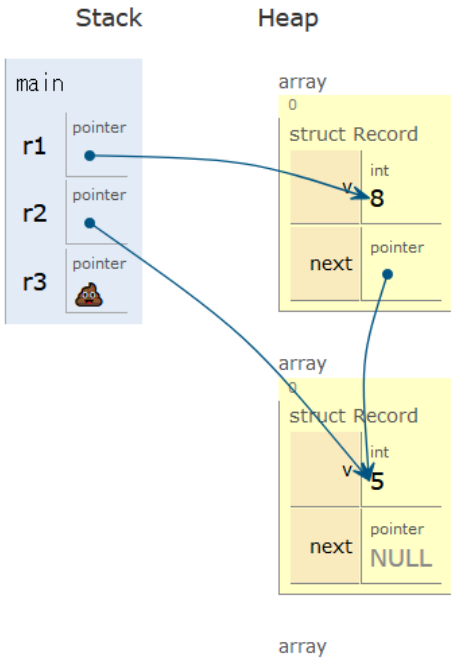
③ 次のようにプログラムを書き換えて，実行し，結果を確認しなさい．**要素を末尾に挿入**している

```
1  #include<stdlib.h>
2
3  struct Record {
4    int v;
5    struct Record *next;
6  };
7
8  int main() {
9    struct Record *r1, *r2, *r3, *r4;
10   r1 = (struct Record *)malloc(sizeof(struct Record));
11   r2 = (struct Record *)malloc(sizeof(struct Record));
12   r3 = (struct Record *)malloc(sizeof(struct Record));
13   r1->v = 8;
14   r1->next = r2;
15   r2->v = 5;
16   r2->next = r3;
17   r3->v = 16;
18   r3->next = NULL;
19   r4 = (struct Record *)malloc(sizeof(struct Record));
20   r4->v = 4;
21   r4->next = NULL;
22   r3->next = r4;
23   return 0;
24 }
```



④ 次のようにプログラムを書き換えて，実行し，結果を確認しなさい．**要素を末尾に挿入**している


```c
#include<stdlib.h>

struct Record {
  int v;
  struct Record *next;
};

int main() {
  struct Record *r1, *r2, *r3;
  r1 = (struct Record *)malloc(sizeof(struct Record));
  r2 = (struct Record *)malloc(sizeof(struct Record));
  r3 = (struct Record *)malloc(sizeof(struct Record));
  r1->v = 8;
  r1->next = r2;
  r2->v = 5;
  r2->next = r3;
  r3->v = 16;
  r3->next = NULL;
  r2->next = NULL;
  free(r3);
  return 0;
}
```



課題

次のリストを作成しなさい

list

0	1	2	3	4
15	8	6	32	23