Задача 4. core DL

Источник:	основная I
Имя входного файла:	`---`
Имя выходного файла:	`stdout`
Ограничение по времени:	разумное
Ограничение по памяти:	разумное

Пусть `State` — это массив из 256 строковых “регистров”:

```
typedef struct State {
    char *regs[256];
} State;
```

Каждый регистр либо нулевой (т.е. указатель равен 0), либо указывает на C-шную строку, расположенную на куче.

Требуется реализовать следующие функции:

```
//prints 'ECHO: ' and all passed strings separated by '|'
void echo_0(State *state);
void echo_1(State *state, char *arg0);
void echo_2(State *state, char *arg0, char *arg1);
void echo_3(State *state, char *arg0, char *arg1, char *arg2);
//prints contents of I-th register (it must not be NULL)
//[idx] contains decimal representation of I
void print_1(State *state, char *idx);
//prints all non-NULL registers with their values (sorted by register number)
void printregs_0(State *state);
//saves a copy of string [what] into I-th register
//[idx] contains decimal representation of I
void store_2(State *state, char *idx, char *what);
//copies contents of S-th register into D-th register (S-th register is not NULL)
//[dst] and [src] contain decimal representations of D and S respectively
//BEWARE: [dst] and [src] are allowed to be equal indices
void copy_2(State *state, char *dst, char *src);
//assigns NULL to I-th register
//[idx] contains decimal representation of I
void clear_1(State *state, char *idx);
```

Замечания:

1. Требуемый формат вывода у функций `echo_*`, `print_1` и `printregs_0` можно посмотреть в примере ниже.
2. Функции `echo_*` хоть и принимают параметр `state`, но его **не** используют.
3. Учтите, что каждый ненулевой регистр должен указывать на собственный буфер памяти, ни с кем не разделённый. Если функция записывает что-то в регистр, то память от старого значения нужно удалить, а память для нового значения нужно выделить.

Для проверки отправьте исходный код и скрипт сборки `c.sh.txt`. Скрипт должен собрать динамическую библиотеку `core.so`.

Код примера:

```
#include <assert.h>
typedef struct State {
```

```
    char *regs[256];
} State;
#include "decls.h"  //contains function declarations (chunk of code above)
State state;
int main() {
    echo_2  (&state, "hello", "world");
    echo_0  (&state);
    echo_1  (&state, "the_only_argument");
    echo_3  (&state, "a", "b", "c");
    store_2 (&state, "13", "thirteen");
    store_2 (&state, "10", "ten");
    store_2 (&state, "15", "fifteen");
    store_2 (&state, "20", "twelve");
    echo_1  (&state, "==state==");
    printregs_0(&state);
    echo_1  (&state, "==copying==");
    print_1 (&state, "13");
    print_1 (&state, "15");
    copy_2  (&state, "13", "15");
    print_1 (&state, "13");
    print_1 (&state, "15");
    echo_1  (&state, "==clear==");
    clear_1 (&state, "10");
    clear_1 (&state, "15");
    store_2 (&state, "13", "thirteen_V2");
    printregs_0(&state);
}
```

В результате запуска в стандартный поток вывода должно быть записано:

```
ECHO: hello|world
ECHO:
ECHO: the_only_argument
ECHO: a|b|c
ECHO: ==state==
10 = ten
13 = thirteen
15 = fifteen
20 = twelve
ECHO: ==copying==
thirteen
fifteen
fifteen
fifteen
ECHO: ==clear==
13 = thirteen_V2
20 = twelve
```