cmp — at most 190 points
(the Scientific Committee only has a solution for 100 points)

Source code: `cmp.c, cmp.cpp, cmp.pas`

Time limit: `10 seconds`

Memory limit: `16 MB in addition to what Committee's library uses`

Трябва да съставите алгоритъм за старинен компютър, чиято памет е един масив от 10240 бита. Паметта е инициализирана с нули. Можете да четете и записвате по един бит за всеки момент от време. Заданието ви е да реализирате две операции за този компютър:

remember(*a*): *a* е цяло число между 0 и 4095
Имплементацията на тази операция може да извиква
bit_set(*address*): *address* е цяло число между 1 и 10240
Битът на позицията *address* става 1

compare(*b*): *b* е цяло число между 0 и 4095
ако *b* < *a*, трябва да върне –1
ако *b* = *a*, трябва да върне 0
ако *b* > *a*, трябва да върне 1
Имплементацията на тази операция може да извиква
bit_get(*address*)
което връща стойността на бита памет от позиция *address*: 1 ако той е бил установен чрез *bit_set*(), когато е изпълнявана операцията *remember*(*a*), или връща 0 в противен случай.

Задача

Имплементирайте *remember*() и *compare*() така, че да минимизирате общия брой на операциите за достъп до паметта (*bit_set*() и *bit_get*()) в най-лошия случай за всички възможни стойности на *a* и *b*.

Вашето решение ще бъде оценявано чрез пълно изчерване:

define AllMemory = array [0..4095][1..10240] of bits
set AllMemory to zeros
for *a* = 0..4095:
define *bit_set*(*address*): AllMemory[*a*][*address*] = 1
***remember*(*a*)**
let *maxA*= the maximum number of *bit_set*() calls executed for any *a*
for (a,b) ∈ {0..4095}×{0..4095} in random order (i.e. all valid pairs (a,b) are considered, in some random order)
define *bit_get*(*address*): return AllMemory[*a*][*address*]
***answer* =*compare*(*b*)**
if *answer* for comparing *a* and *b* is incorrect : your score = 0; exit
let *maxB* = the maximum number of *bit_get()* calls executed for any (*a*,*b*) pair
T*=*maxA* + *maxB
If (*T*>20): your score = 0; exit
else your score = 1 + 9 * (21– *T*); exit

Описание на имплементацията

Имплементация на C:

Вашият сорс трябва да започва с:

```
#include "cmp.h"
```

Прототипите на фунциите за достъп до паметта са:

```
void bit_set(int addr);
int bit_get (int addr);
```

Вие трябва да имплементирате фунциите:

```
void remember(int value);
int compare(int value);
```

Файлът *cmp.h* ще осигури функцията *main*() и вие трябва да не дефинирате друга такава функция. Всички глобални имена (за променливи и фунции) с изключение на *bit_set*() и *bit_get*() ще започват с префикс *boi*. Ако не именувате вашите променливи и функции с този префикс, ще избегнете грешки при компилацията,

За ваше удобство ще ви бъде предоставена примерна имплементация за *cmp.h* във файла *public-cmp.h*, която вие може да редактирате, ако желаете. Тази имплементация е различна от версията *cmp.h* , която ще се използва за оценяване, но интерфейсът е същия.

За да компилирате целия код, компилирайте вашия сорс като поставите *cmp.h* в същата директория.

Изтегляне на файлове:

Кликнете върху "*download task description*" на уеб страницата за тази задача, за да изтеглите архива *cmp.zip*. Този архив съдържа *cmp.h*, *cmpdata.pas* и *public-cmp.pas,* които трябва да използвате за вашете решение, а също и описанието на задачата, и примерните имплементации *cmp.pas*, *cmp.c*, *cmp.cpp* (файловете с окончание *pas* са за програмиращите на Паскал).

Ограничения

- Ще бъдете дисквалифицирани при всеки опит да взаимодействате с паметта, използвана от кода за оценяване на журито.
- Ако вашият код не изпълнява дефинирания по-горе протокол (например извиква *bit_set*() по време на *compare*() или извиква невалидни адреси), ще получите 0 точки.
- **Time limit:** вашата реализация трябва да изпълни 4096 извиквания на *remembe*r() и 4096*4096 извиквания на *compare*() в рамките на 10 секунди.