Задача №1

Намиране на максимум

(50 точки)

Алгоритъм

Всеки знае как да намери най-големия елемент в масив от цели числа. За тази цел може да се използва следният (псевдо) код:

```
max = x[0];
for (i = 1; i < n; i++){
     if (x[i] > max)
          max = x[i];
}
```

Този фрагмент от код стъпва върху предположението, че разполагаме с един централен процесор, който използва **(n-1)** сравнения за да намери най-големия елемент в масива. Възможно ли е задачата да се реши по-бързо при положение, че разполагаме с повече процесори? Един възможен подход е описаният по-долу.

Да предположим, че са ни дадени **n** различни цели числа **x[0], x[1], ..., x[n-1]**.

1. **Стъпка 1:** Инициализираме работен масив **w[]** от **n** елемента така, че всеки от елементите му да има начална стойност **1**. За тази цел могат да се използват **n** нишки, като всяка нишка записва **1** в съответстващият и елементи. По-точно, нишката **T(i)** записва **1** в елемента **w[i]**. Ако разполагаме с **n** процесора, то тази стъпка би се изпълнила (в идеалния случай) за една стъпка.

2. **Стъпка 2:** За всяка двойка цели числа **x[i]** и **x[j]** се създава нишка **T(i,j)**. Тази нишка сравнява числата **x[i]** и **x[j]** и записва 0 в **w[i]** ако **x[i]<x[j]**. В противен случай нишката **T(i,j)** записва **0** в елемента **w[j]**. За тази стъпка са необходими **n*(n-1)/2** нишки, всяка от които изпълнява едно сравнение и едно присвояване. (Защо са необходими **n*(n-1)/2** нишки, а не **n^2**?)

3. **Стъпка 3:** Тази стъпка изисква **n** нишки. Нишката **T(i)** проверява стойността на **w[i]**. Ако стойността е **0**, то нишката не прави нищо. Ако стойността е **1**, то тази нишка отпечатва стойността на **x[i]**, защото това е най-големият елемент на масива **x[]**!

   Причината за това е следната: най-голяма стойност на масива **x[]** е по-голяма от всички останали. Поради тази причина при изпълнение на **Стъпка 2** съответният елемент в масива **w[]** никога няма да получи стойност нула.

   За изпълнението на Стъпка 3 са необходими **n** нишки, като всяка нишка изпълнява по едно сравнение.

Пример

Нека разгледаме следния пример. Да предположим, че имаме четири различни цели числа, които са зададени като стойности на масива **x[]**: **x[]={3,1,7,4}.** Всички елементи на масива **w[]** се инициализират със стойност **1**: **w[]={1,1,1,1}**. За изпълнението на **Стъпка 2** имаме нужда от **n*(n-1)/2=4*(4-1)/2=6** нишки. Резултатите от изпълнението на **Стъпка 2** са представени в следната таблица:

Нишка	Сравнение	Операция
T(0,1)	Сравнява **x[0]<x[1]**, т.е. **3<1**	**w[1]=0**
T(0,2)	Сравнява **x[0]<x[2]**, т.е. **3<7**	**w[0]=0**
T(0,3)	Сравнява **x[0]<x[3]**, т.е. **3<4**	**w[0]=0**
T(1,2)	Сравнява **x[1]<x[2]**, т.е. **1<7**	**w[1]=0**
T(1,3)	Сравнява **x[1]<x[2]**, т.е. **1<4**	**w[1]=0**
T(2,3)	Сравнява **x[2]<x[3]**, т.е. **7<4**	**w[3]=0**

След изпълнението на **Стъпка 2** стойностите на елементите **w[0]**, **w[1]** и **w[3]** са равни на нула, а стойността на елемента **w[2]** все още е **1**. Следователно в **Стъпка 3** нишката, асоциирана с елемента **w[2]** ще изведе стойността на елемента **x[2]**, която е **7**. Следователно максималният елемент в масива **x[]** е **7**.

Вход и изход

Програмата трябва да получи данните си от командния ред. Изпълнимият файл трябва да се казва **prog1**. Командният ред трябва да изглежда по следния начин:

prog1 n x[0] x[1] x[2] ... x[n-1]

Командният ред започва с името на изпълнимия файл **prog1**, последвано от цяло число **n**, указващо броя на целите числа, които ще се сравняват, последвано от съответния брой различни цели числа. Можете да приемете, че броят на целите числа ще бъде в диапазона от **3** до **100**.

По време на работата си вашата програма трябва да извежда следната информация:

- Стойностите на масива **w[]** след стъпката по инициализация на масива — Стъпка 1.
- Действията извършвани от нишката **T(i,j)** по време на изпълнение на Стъпка 2.
- Стойностите на елементите на масива **w[]** след изпълнението на Стъпка 2.
- Стойността на най-големия елемент в масива **x[]** и неговото местоположение (индекс).

Да предположим, че командният ред има следния вид:

prog1 4 3 1 7 4

Тогава изходът от вашата програма трябва да има следният вид:

```
Number of input values = 4
Input values          x = 3 1 7 4
After initialization w = 1 1 1 1
          ..........
Thread T(1,3) compares x[1]=1 and x[3]=4, and writes 0 into w[1]
```

```
        ..........
After Step 2          w = 0 0 1 0
Maximum                 = 7
Location                = 2
```

Изисквания към решението

1. Програмата трябва да бъде написана на езика **C++** съгласно **ISO/IEC 14882:1998**.
2. Задължително към файловете с решението да е приложен и **Makefile**. Изпълнимият файл, който се създава по време на компилация на решението, трябва да се казва **prog1**.
3. При проверка на решението програмата ви ще бъде компилирани и тествана по следния начин:
   ```
   make
   prog1 4 3 1 7 4
   ```
   Предходната процедура ще бъде изпълнена няколко пъти с различни входни данни за да се провери дали вашата програма работи коректно.
4. Реализацията на програмата трябва да спазва точно изискванията описани по-горе. Всяко отклонение от изискванията ще доведе до получаване на 0 точки. По-точно: използвайте **n** нишки за инициализация в **Стъпка 1**, **n*(n-1)/2** нишки за **Стъпка 2** и **n** нишки за **Стъпка** 3.
5. През цялото време трябва да се стремите да постигнете възможно най-голям паралелизъм. Реализирайте решението както е описано по-горе и се погрижете нишките да могат да се изпълняват едновременно във всяка от стъпките на решението.
6. Работи, които са предадени по-късно от обявеното (или не са предадени), ще бъдат оценени с 0 точки.
7. Правилата за оценяване са следните. Приемаме, че напълно коректна и написана спрямо изискванията програма получава максималния брой точки — 100%. Ако в решението има пропуски, максималният брой точки ще бъде намален съгласно следните правила:
   - Програмата ви трябва да съдържа достатъчно коментари. Оценката на решения без коментари или с недостатъчно и/или мъгляви коментари ще бъде намалена с 30%.
   - Всеки файл от решението трябва да започва със следният коментар:
     ```
     //-----------------------------------------
     // NAME: Ivan Ivanov
     // CLASS: Xia
     // NUMBER: 13
     // PROBLEM: #1
     // FILE NAME: xxxxxx.yyy.zzz (unix file name)
     // FILE PURPOSE:
     //   няколко реда, които описват накратко
     //   предназначението на файла
     //-----------------------------------------
     ```
   - Всяка функция във вашата програма трябва да включва кратко описание в следния формат:
     ```
     //-----------------------------------------
     // FUNCTION: xxyyzz (име на функцията)
     //   предназначение на функцията
     // PARAMETERS:
     //   списък с параметрите на функцията
     ```

```
//   и тяхното значение
// FUNCTIONS CALLED:
//   списък на функциите, които се извикват
//   от описваната функция
//--------------------------------------------
```

- Лош стил на програмиране и липсващи заглавни коментари ще ви костват 30%.
- Програми, които не се компилират получават 0 точки. Под „**не се компилират**“ се има предвид произволна причина, която може да причини неуспешна компилация, включително липсващи файлове, неправилни имена на файлове, синтактични грешки, неправилен или липсващ **`Makefile`**, и т.н. Обърнете внимание, че в UNIX имената на файловете са case sensitive.
- Програми, които се компилират, но не работят, не могат да получат повече от 50%. Под „**компилира се, но не работи**“ се има предвид, че вие сте се опитали да решите проблема до известна степен, но не сте успели да направите пълно решение. **Безсмислени или мъгляви програми ще бъдат оценявани с 0 точки, независимо че се компилират**. Често срещан проблем, който спада към този случай, е че вашият **`Makefile`** генерира изпълним файл, но той е именуван с име, различно от очакваното (т.е. **`prog1`** в разглеждания случай).
- Програми, които дават неправилни или непълни резултати, или програми, в които изходът и/или форматирането се различава от изискванията ще получат не повече от 70%.
- Всички наказателни точки се сумират. Например, ако вашата програма няма задължителните коментари в началото на файлове и функциите се отнемат 30%, ако няма достатъчно коментари се отнемат още 30%, компилира се, но не работи правилно — още 30%, то тогава резултатът ще бъде:
  `50*(100-30-30-30)%= 50*10%= 5 точки`.
- Работете самостоятелно. Групи от работи, които имат твърде много прилики една с друга, ще бъдат оценявани с 0 точки.
- Вашата програма трябва да съдържа следните файлове:
  1. Заглавен файл **`MaxThread.hh`**, който съдържа дефинициите на всички класове, наследници на класа **`Thread`**, които са необходими за решението на задачата.
  2. Файл **`MaxThread.cc`**, който съдържа всички имплементации на класовете, дефинирани в **`MaxThread.hh`**.
  3. Файл **`main.cc`**, в който е дефинирана главната функция на програмата **`main()`**.
  4. Набора от файлове **`Thread.hh`**, **`Thread.cc`**, **`Mutex.hh`**, **`Mutex.cc`**, **`Guard.hh`**, **`ThreadError.hh`**, **`ThreadError.cc`**, разработвани по време на упражненията.
  5. Файл **`Makefile`**, който компилира описаните по-горе файлове и създава изпълним файл с име **`prog1`**. Обърнете внимание, че ако вашето решение не съдържа **`Makefile`** или съдържа некоректен **`Makefile`**, то най-вероятно ще попадне в категорията „**не се компилира**“.