Quase Primos Malucos

Nina é uma esperta garota que adora fazer contas. Recentemente Nina pensou em um joguinho muito interessante a qual chamou de `Quase Primos Malucos`.

A ideia do problema dos quase primos malucos que é um número **não** pode ser primo, no entanto todos os divisores (além de 1) devem ser maiores que 10 e devem haver mais de 10 divisores. Também é importante perceber que os divisores devem ser menores que a raiz quadrada do número `quase primo maluco`.

A brincadeira de Nina consiste em *falar* um número qualquer e a outra pessoa ter que responder o menor número *quase primo maluco* estritamente maior que o número dito.

Por exemplo:

Se Nina gritar 1, a resposta deve ser 508079, pois é o menor número maior que 1 que não é primo e possui ao menos 10 divisores maiores que 10, que são: `11 13 17 19 121 143 187 209 221 247 323`. E o mesmo ocorre para todos os números entre 1 e 508079.

Para 600000 o menor `quase primo maluco` é 600457, com os divisores `11 13 17 19 143 169 187 209 221 247 323`.

O Número 26741 não é um número `quase primo maluco` pois os seus divisores são `1 11 13 17 121 143 187 221 1573 2057 2431` e sua raiz quadrada é 163,5, logo todos os divisores maiores que 163 devem ser desconsiderados.

Nina percebeu que sua brincadeira é difícil, pois os números `quase primos malucos` são muito grandes. No entanto ela gostaria de saber de antemão vários números quase primos e pediu a sua ajuda para escrever um programa que seja capaz de responder as questões para ela.

Entrada

A primeira linha contém o inteiro T (1 \leq T \leq 1000), que representa a quantidade de casos de teste.

Cada uma das próximas T linhas contém um número n (1 \leq n \leq 10^9).

Saída

Para cada caso de teste, imprima uma linha contento o menor número `quase primo maluco` que seja estritamente maior que n.

TAREFA

Você já deve ter percebido que esse problema talvez seja melhor ser implementado utilizando threads. Mas tome Cuidado! Você deve imprimir a resposta na ordem relativa a entrada. Ou seja, a resposta nunca pode ser diferente da mostrada nos exemplos abaixo.

Uma proposta para resolver o problema com threads, segue em pseudo-código abaixo:

```
struct parametro_thread
{
  int n;
  int tid;
  int result;
};
int main(void)
{
  leia(QUANTIDADE_DE_CASOS) //só para jogar fora mesmo
  while(1)
  {
    if(leia(n)== EOF) break;
    struct parametro_thread PARAMETRO_A.n=n;
    cria_thread(calcule_sequaseprimo_maluco(PARAMETRO_A))

    if(leia(n)== EOF) break;
    struct parametro_thread PARAMETRO_B.n=n;
    cria_thread(calcule_sequaseprimo_maluco(PARAMETRO_B))

    espera_thread1();
    espera_thread2();
    imprime(PARAMETRO_A.result);
    imprime(PARAMETRO_B.result);

  }
}
```

- o pseudo-código acima possui um problema quando a entrada é ímpar! Tome cuidado.
- o pseudo-código ilustrado acima é somente um exemplo e pode ser melhorado!
- **Use no máximo 2 threads (além da principal)** pois o juiz disponibilizará apenas 2 núcleos de processamento.

Exemplos

Exemplo de entrada

```
2
550794
```

```
6530430
```

Exemplo de saída

```
600457
6533033
```

Exemplo de entrada

```
11
1
2
7
10
11
22
23
123
173
233
2393
```

Exemplo de saída

```
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
```

Exemplo de entrada

```
52
550794
6530430
7664038
8734648
266286
4628267
```

4850022
2069925
9584058
8975573
9483668
3299048
9411688
5364694
5927313
7637634
2579411
8693163
3630866
373379
3146119
4130535
7840298
9081058
2514672
5931337
439326
9520054
409763
5262060
5961227
1718484
5908768
9440313
1461901
3550983
385489
3477510
6455681
6397209
2375701
8372742
6423830
9007066
6176284
9153009
7312751
9091366
6308791
1991077
8900988
9976196

Exemplo de saída

```
600457
6533033
7667803
8735441
508079
4630769
4858243
2070107
9587201
8983871
9484553
3308987
9412117
5370079
5931211
7637641
2582827
8696129
3638063
508079
3149003
4132271
7841977
9081553
2520947
5933719
508079
9520159
508079
5262653
5963243
1733303
5909189
9442259
1466641
3555409
508079
3478387
6455801
6407731
2379949
8381087
6424759
9007603
```

```
6176797
9153287
7315627
9092369
6311591
1994707
8902333
9977147
```

Exemplo de entrada

```
78
2
3
5
7
23
29
31
37
53
59
71
73
79
233
239
293
311
313
317
373
379
593
599
719
733
739
797
2333
2339
2393
2399
2939
3119
3137
```

3733
3739
3793
3797
5939
7193
7331
7333
7393
23333
23339
23399
23993
29399
31193
31379
37337
37339
37397
59393
59399
71933
73331
73939
233993
239933
293999
373379
373393
593933
593993
719333
739391
739393
739397
739399
2339933
2399333
2939999
3733799
5939333
7393913
7393931
7393933

Exemplo de saída

508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079

```
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
508079
600457
600457
726869
775489
775489
775489
775489
2345057
2404259
2951897
3736447
5943223
7395949
7395949
7395949
```

Author: Bruno Ribas, inspirado no problema ‘Almost Prime Numbers’ do Topcoder