Nome:			RA:				
Assinatura:	1	2	3	4			
MC102 – GABARITO Prova 1 2009a Prof. Rogério Drummond	2	3	3	3			

Use lapis de preferência. Desenhe um retângulo envolta de cada resposta, identificando claramente a questão respondida. Só serão consideradas as respostas nos locais indicados. Nenhum outro papel além desta prova é permitido durante a aplicação deste teste. Calculadoras, telefones, radios, etc também não são permitidos.

```
// Requer x>1, retorna 1 se x eh primo, 0 caso contrario
int primo(int x) { … }
```

1. Implemente a função `int medio3(int a, int b, int c)` que retorna o valor do meio dentre os parâmetros. Os 3 parâmetros tem valores diferentes

```
int medio3(int a, int b, int c) {
   if (a<b && a>c || a>b && a<c)
      return a;
   if (b<a && b>c || b>a && b<c)
      return b;
   return c;
}
```

```
// versao usando maior3() e menor3()
int medio3(int a, int b, int c) {
   if (a!=maior3(a,b,c) && a!=menor3(a,b,c))
      return a;
   if (b!=maior3(a,b,c) && b!=menor3(a,b,c))
      return b;
   return c;
}
```

2. Dado a implementação da raiz quadrada abaixo, estime quantas vezes o comando `if` do `while` é executado (em função dos valores de `x` e `p`). Justifique sua resposta.

```
// pressupoe x>1 e p>0
long double raiz2(long double x, long double p) {
   long double inicio, fim, raiz;
   inicio = 1;
   fim = x;
   raiz = (inicio + fim) / 2;
   while ((fim - inicio) / 2 >= p) {  // veja observacoes abaixo
      if (raiz*raiz < x)
         inicio = raiz;
      else
         fim = raiz;
      raiz = (inicio + fim) / 2;
   }
}
```

O while para quando metade do intervalo de busca for menor que p. A cada interação do while o intervalo é dividido ao meio.
O intervalo inicial é [1,x], resta saber quantas vezes (x-1) pode ser dividido por 2 até que fique menor que p.
Analiticamente:
(x-1) / 2^(k+1) < p ➔ 2^(k+1) > (x-1) / p, aplicando $\log_2$
➔ k+1 > $\log_2$ ((x-1) / p). Entao o while para quando: k >= $\log_2$ ((x-1)/p) - 1

3. Escreva a função `int proximoPrimo(int n)` que retorna o menor número primo maior que `n`. Você pode usar a função `int primo(int x)` na sua solução. Eficiência e clareza serão considerados na avaliação.

```
int proximoPrimo(int n) {
   n = n + 1;
   if (n==2)
      return 2;
   if ( n%2 == 0 )
      n = n + 1;
   while (primo(n) == 0)
      n = n + 2;
   return n;
}
```

```
// versao compacta
int proximoPrimo(int n) {
   n++;
   if (n==2)
      return 2;
   if ( !(n%2) )
      n++;
   while (! primo(n)) n+=2;
   return n;
}
```

4. Escreva a função `void fator(int n)` que imprime os fatores primos de `n` no formato de um produtório de `p^k`, onde `p` é o número primo. Por exemplo 180 = 2^2 x 3^2 x 5^1. Você pode usar a função `int proximoPrimo(int x)` na sua solução. Eficiência e clareza serão considerados na avaliação.

```
void fator(int n) {
   int p = 2, k;
   printf("\n%d = 1", n);
   while (n>1) {
      if (n%p == 0) {
         k = 0;
         do {
            n = n/p;
            k = k + 1;
         } while (n%p == 0);
         printf(" x %d^%d", p, k);
      }
      p = proximoPrimo(p);
   }
   printf("\n");
}
```

```
// versao compacta
void fator(int n) {
   int p = 2, k;
   printf("\n%d = 1", n);
   while (n>1) {
      if (! (n%p)) {
         k = 0;
         do {
            n /= p;
            k++;
         } while (!n%p);
         printf(" x %d^%d", p, k);
      }
      p = proximoPrimo(p);
   }
   printf("\n");
}
```