

```

#include <stdio.h>
#include <string.h>
#include <sys/time.h>
#include <unistd.h>
#include <sys/resource.h>
#include <malloc.h>

int count=0;

int timeval_subtract (result,x,y)
struct timeval *result, *x, *y;
{
    if (y->tv_usec < x->tv_usec) {
        result->tv_usec = 1000000 - (x->tv_usec - y->tv_usec);
        result->tv_sec = y->tv_sec - x->tv_sec - 1;
    } else {
        result->tv_usec = y->tv_usec - x->tv_usec;
        result->tv_sec = y->tv_sec - x->tv_sec;
    }

    return result->tv_sec*1000000 + result->tv_usec;
}

const int NumIt = 11;
int MaxNum=8191;

main(argc,argv)
int argc;
char *argv[];
{
    struct timeval result,tv0, tv1;
    int n,i,k,it, NumPrimes;
    int *isPrime;

    isPrime= (int*) malloc(sizeof(int)*(MaxNum*(1<<NumIt)+1));

    for (it=1;(it<=NumIt);it++) {

        for (i=1;i<=MaxNum;i++) {
            isPrime[i]=1;
        }
        i=2;

        gettimeofday(&tv0,NULL);

        while (i*i<=MaxNum) {
            if (isPrime[i]) {
                k=i*i;
                while (k<=MaxNum) {
                    isPrime[k]=0;
                    k+=i;
                }
            }
            i++;
        }
        gettimeofday(&tv1,NULL);

        NumPrimes=0;

        for (i=1;i<=MaxNum;i++) {
            if (isPrime[i]) {
                NumPrimes++;
            }
        }

        printf("Array Size %d kB, Nof Primes: %d in %d usecs\n",
            MaxNum*4/1024+1,NumPrimes,timeval_subtract(&result, &tv0, &tv1));

```

```

        MaxNum*=2;
    }
}

Ultra-1, Model 140 UltraSPARC, 64-Bit RISC Processor
SunOS Release 5.6
Array Size 32 kB, Nof Primes: 1029 in 264 usecs
Array Size 64 kB, Nof Primes: 1901 in 494 usecs
Array Size 128 kB, Nof Primes: 3513 in 990 usecs
Array Size 256 kB, Nof Primes: 6543 in 3213 usecs
Array Size 512 kB, Nof Primes: 12249 in 12621 usecs
Array Size 1024 kB, Nof Primes: 22997 in 90798 usecs
Array Size 2048 kB, Nof Primes: 43385 in 221921 usecs
Array Size 4096 kB, Nof Primes: 82020 in 489097 usecs
Array Size 8192 kB, Nof Primes: 155592 in 1'039777 usecs
Array Size 16383 kB, Nof Primes: 295917 in 2'182027 usecs
Array Size 32765 kB, Nof Primes: 564108 in 6'765513 usecs
Array Size 65529 kB, Nof Primes: 1077744 in 11'352244 usecs

Ultra-5_10, 333 MHz, UltraSPARC-III, 64-Bit RISC Processor
SunOS Release 5.6
Array Size 32 kB, Nof Primes: 1029 in 114 usecs
Array Size 64 kB, Nof Primes: 1901 in 270 usecs
Array Size 128 kB, Nof Primes: 3513 in 454 usecs
Array Size 256 kB, Nof Primes: 6543 in 872 usecs
Array Size 512 kB, Nof Primes: 12249 in 4557 usecs
Array Size 1024 kB, Nof Primes: 22997 in 8460 usecs
Array Size 2048 kB, Nof Primes: 43385 in 26387 usecs
Array Size 4096 kB, Nof Primes: 82020 in 393431 usecs
Array Size 8192 kB, Nof Primes: 155592 in 995271 usecs
Array Size 16383 kB, Nof Primes: 295917 in 2'220950 usecs
Array Size 32765 kB, Nof Primes: 564108 in 4'693115 usecs
Array Size 65529 kB, Nof Primes: 1077744 in 9'858742 usecs

Pentium Pro, 200 MHz, 32-Bit CISC Processor
Linux 2.2.7
Array Size 32 kB, Nof Primes: 1029 in 660 usecs
Array Size 64 kB, Nof Primes: 1901 in 1438 usecs
Array Size 128 kB, Nof Primes: 3513 in 3179 usecs
Array Size 256 kB, Nof Primes: 6543 in 14063 usecs
Array Size 512 kB, Nof Primes: 12249 in 54725 usecs
Array Size 1024 kB, Nof Primes: 22997 in 136323 usecs
Array Size 2048 kB, Nof Primes: 43385 in 303678 usecs
Array Size 4096 kB, Nof Primes: 82020 in 635517 usecs
Array Size 8192 kB, Nof Primes: 155592 in 1'327639 usecs
Array Size 16383 kB, Nof Primes: 295917 in 2'738676 usecs
Array Size 32765 kB, Nof Primes: 564108 in 6'461780 usecs
Array Size 65529 kB, Nof Primes: 1077744 in 12'944238 usecs

Pentium III, 500 MHz, 32-Bit CISC Processor
Linux 2.2.7
Array Size 32 kB, Nof Primes: 1029 in 575 usecs
Array Size 64 kB, Nof Primes: 1901 in 5475 usecs
Array Size 128 kB, Nof Primes: 3513 in 3069 usecs
Array Size 256 kB, Nof Primes: 6543 in 9398 usecs
Array Size 512 kB, Nof Primes: 12249 in 26493 usecs
Array Size 1024 kB, Nof Primes: 22997 in 70144 usecs
Array Size 2048 kB, Nof Primes: 43385 in 171306 usecs
Array Size 4096 kB, Nof Primes: 82020 in 348877 usecs
Array Size 8192 kB, Nof Primes: 155592 in 789302 usecs
Array Size 16383 kB, Nof Primes: 295917 in 1559357 usecs
Array Size 32765 kB, Nof Primes: 564108 in 3280360 usecs
Array Size 65529 kB, Nof Primes: 1077744 in 6714938 usecs

```

```

#include <stdio.h>
#include <string.h>
#include <sys/time.h>
#include <unistd.h>
#include <sys/resource.h>

int count;

int timeval_subtract (result,x,y)
    struct timeval *result, *x, *y;
{
    if (y->tv_usec < x->tv_usec) {
        result->tv_usec = 1000000 - (x->tv_usec - y->tv_usec);
        result->tv_sec = y->tv_sec - x->tv_sec - 1;
    } else {
        result->tv_usec = y->tv_usec - x->tv_usec;
        result->tv_sec = y->tv_sec - x->tv_sec;
    }

    return result->tv_sec*1000000 + result->tv_usec;
}

/* Ackermann Function as defined in Text Book */
int ackermann(int m, int n)
{
    /*count++;*/
    if (m==0) return (n+1);
    else {
        if (n==0) return ackermann((m-1),1);
        else return ackermann((m-1), ackermann(m,(n-1)));
    }
}

/* Ackermann Function as defined in Lecture Notes
   Recursion Hypothesis is the same but Anchoring is slightly different
   The following formula shows the relation between the 2 definitions:
   ack(m,n) = ack2(m-2,m+n)-3
*/
int ackermann2(int m, int n)
{
    /*count++;*/
    if (m==1) return (1<n);
    else {
        if (n==1) return ackermann2((m-1),1);
        else return ackermann2((m-1), ackermann2(m,(n-1)));
    }
}

main(argc,argv)
    int argc;
    char *argv[];
{
    int m,n,k,k1,j,j2;
    struct timeval result,tv0, tv1;
    int num;

    k=16;
    k1=1;
    m=3;

    for (n=1;n<=8;n++) {
        count=0;

        gettimeofday(&tv0,NULL);
        j=ackermann(m,n);
        gettimeofday(&tv1,NULL);

```

```

/* just to test that ack2 delivers the same results */
count=0;
j2=ackermann2(m-2,m+n)-m;
if (j!=j2) printf("Wrong Value ack=%d, ack2=%d/n",j,j2);

num=(512*k1-15*k+9*n+37)/3;
printf("Ackermann(%d,%d)=%d: ( %d proc.calls in %d usecs)\n",
        m,n,j,num,timeval_subtract(&result, &tv0, &tv1));
k1*=4;
k*=2;
}
}

Ultra-1, Model 140 UltraSPARC, 64-Bit RISC Processor
SunOS Release 5.6
Ackermann(3,1)=13: ( 106 proc.calls in 21 usecs)
Ackermann(3,2)=29: ( 541 proc.calls in 145 usecs)
Ackermann(3,3)=61: ( 2432 proc.calls in 856 usecs)
Ackermann(3,4)=125: ( 10307 proc.calls in 4169 usecs)
Ackermann(3,5)=253: ( 42438 proc.calls in 18069 usecs)
Ackermann(3,6)=509: ( 172233 proc.calls in 76802 usecs)
Ackermann(3,7)=1021: ( 693964 proc.calls in 305451 usecs)
Ackermann(3,8)=2045: ( 2785999 proc.calls in 1232650 usecs)

Ultra-5_10, 333 MHz, UltraSPARC-III, 64-Bit RISC Processor
SunOS Release 5.6
Ackermann(3,1)=13: ( 106 proc.calls in 9 usecs)
Ackermann(3,2)=29: ( 541 proc.calls in 71 usecs)
Ackermann(3,3)=61: ( 2432 proc.calls in 379 usecs)
Ackermann(3,4)=125: ( 10307 proc.calls in 1947 usecs)
Ackermann(3,5)=253: ( 42438 proc.calls in 7580 usecs)
Ackermann(3,6)=509: ( 172233 proc.calls in 35460 usecs)
Ackermann(3,7)=1021: ( 693964 proc.calls in 148316 usecs)
Ackermann(3,8)=2045: ( 2785999 proc.calls in 606249 usecs)

Pentium Pro 200 MHz, 32-Bit CISC Processor
Linux 2.0.30
Ackermann(3,1): ( 106 proc.calls) in 14 usecs
Ackermann(3,2): ( 541 proc.calls) in 49 usecs
Ackermann(3,3): ( 2432 proc.calls) in 129 usecs
Ackermann(3,4): ( 10307 proc.calls) in 499 usecs
Ackermann(3,5): ( 42438 proc.calls) in 1992 usecs
Ackermann(3,6): ( 172233 proc.calls) in 8301 usecs
Ackermann(3,7): ( 693964 proc.calls) in 38158 usecs
Ackermann(3,8): ( 2785999 proc.calls) in 169501 usecs

Pentium III 500 MHz, 32-Bit CISC Processor
Linux 2.2.7
Ackermann(3,1)=13: ( 106 proc.calls in 5 usecs)
Ackermann(3,2)=29: ( 541 proc.calls in 13 usecs)
Ackermann(3,3)=61: ( 2432 proc.calls in 50 usecs)
Ackermann(3,4)=125: ( 10307 proc.calls in 200 usecs)
Ackermann(3,5)=253: ( 42438 proc.calls in 4227 usecs)
Ackermann(3,6)=509: ( 172233 proc.calls in 3261 usecs)
Ackermann(3,7)=1021: ( 693964 proc.calls in 17206 usecs)
Ackermann(3,8)=2045: ( 2785999 proc.calls in 85507 usecs)

```