```c
/* Program to fill cache. */
#include <stdio.h>
#include <malloc.h>

#define LG_LINE_SIZE 4
#define LG_SET_COUNT 10
#define ASSOCIATIVITY 4

#define SET_COUNT (1<<LG_SET_COUNT)
#define CACHE_SIZE ( ASSOCIATIVITY << ( LG_LINE_SIZE + LG_SET_COUNT ) )

char a[ 2 * CACHE_SIZE ];


int main(int argv, char **argc)
{
  int dummy = 0;

  printf("Cache parameters:\n");
  printf("  %d-way set associative, 2^%d = %d sets.\n",
         ASSOCIATIVITY, LG_SET_COUNT, 1<<LG_SET_COUNT);
  printf("  Line size 2^%d = %d characters, character size 8 bits = 1 byte.\n",
         LG_LINE_SIZE, 1<<LG_LINE_SIZE);
  printf("  Cache capacity %d bytes, %.2f MiB (megibytes) [sic].\n",
         ASSOCIATIVITY << ( LG_LINE_SIZE + LG_SET_COUNT ),
         ((double)(ASSOCIATIVITY << ( LG_LINE_SIZE + LG_SET_COUNT )))
         / ( 1 << 20 ));


```

```
29
30    /* Fill up cache with minimum number of accesses. */
31
32    {
33      int s,w;
34      char *addr = &a[0];
35      int linesize = 1 << LG_LINE_SIZE;
36
37      for(w=0; w<ASSOCIATIVITY; w++)
38
39        for(s=0; s<SET_COUNT; s++)
40          {
41            dummy += *addr;
42            /* Below, increment index part of address. */
43            addr += linesize;
44          }
45    }
46
```

```
46
47  /* Fill up cache with minimum number of accesses. (Alternate method.) */
48
49  {
50    int w;
51
52    for(w=0; w<ASSOCIATIVITY; w++)
53      {
54        char *b = & a[ w << ( LG_LINE_SIZE + LG_SET_COUNT ) ];
55        int s;
56
57        for(s=0; s<SET_COUNT; s++) dummy += b[ s << LG_LINE_SIZE ];
58      }
59  }
60
```

```
60
61    /* Generate lots of misses while accessing only ASSOCIATIVITY + 1
62       array elements. */
63
64    {
65      int i, w;
66      int deadly_stride = 1 << ( LG_LINE_SIZE + LG_SET_COUNT );
67
68      for(i=0; i<100; i++)
69        {
70          char *addr = &a[0];
71
72          for(w=0; w<=ASSOCIATIVITY; w++)
73            {
74              dummy += *addr;
75              /* Below: Increment tag part of address. */
76              addr += deadly_stride;
77            }
78        }
79    }
80
81    if( dummy )
82      printf("This line prevents optimizer from eliminating code above.\n");
83
84    return 0;
85  }
end
```