28. Developing a client that contacts a given DNS server to resolve a given hostname in java/C.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#ifdef _WIN32
   #include <winsock2.h>
   #include <ws2tcpip.h>
   #pragma comment(lib, "ws2_32.lib")
#else
   #include <sys/types.h>
   #include <sys/socket.h>
   #include <netdb.h>
   #include <unistd.h>
#endif

void resolve_hostname(const char *hostname) {
#ifdef _WIN32
   WSADATA wsa;
   if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {
      printf("WSAStartup failed\n");
      return;
   }
#endif

   struct addrinfo hints, *res, *p;
   char ipstr[INET6_ADDRSTRLEN];

   memset(&hints, 0, sizeof hints);
   hints.ai_family = AF_UNSPEC; // Support both IPv4 & IPv6
   hints.ai_socktype = SOCK_STREAM;

   if (getaddrinfo(hostname, NULL, &hints, &res) != 0) {
      printf("Could not resolve hostname: %s\n", hostname);
#ifdef _WIN32
      WSACleanup();
#endif
      return;
   }

   printf("IP addresses for %s:\n", hostname);
   for (p = res; p != NULL; p = p->ai_next) {
```

```
        void *addr;
        if (p->ai_family == AF_INET) { // IPv4
            struct sockaddr_in *ipv4 = (struct sockaddr_in *)p->ai_addr;
            addr = &(ipv4->sin_addr);
        } else { // IPv6
            struct sockaddr_in6 *ipv6 = (struct sockaddr_in6 *)p->ai_addr;
            addr = &(ipv6->sin6_addr);
        }
        inet_ntop(p->ai_family, addr, ipstr, sizeof ipstr);
        printf("  %s\n", ipstr);
    }

    freeaddrinfo(res);

#ifdef _WIN32
    WSACleanup();
#endif
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Usage: %s <hostname>\n", argv[0]);
        return 1;
    }
    resolve_hostname(argv[1]);
    return 0;
}
```