```
(base) andre@DESKTOP-UM1B7BM:/mnt/c/Users/andre/OneDrive/Systems/Lab06$ make
cc -Wall -g   -c -o list.o list.c
cc -o list list.o
(base) andre@DESKTOP-UM1B7BM:/mnt/c/Users/andre/OneDrive/Systems/Lab06$ ./list www.uoit.ca http
IPv4 stream port: 80 address: 205.211.180.33
(base) andre@DESKTOP-UM1B7BM:/mnt/c/Users/andre/OneDrive/Systems/Lab06$ ./list www.google.ca http
IPv4 stream port: 80 address: 172.217.165.3
IPv6 stream port: 80 address: ::2607:f8b0:400b:80f:0:0
(base) andre@DESKTOP-UM1B7BM:/mnt/c/Users/andre/OneDrive/Systems/Lab06$ ./list science.uoit.ca
Usage: list hostname service
(base) andre@DESKTOP-UM1B7BM:/mnt/c/Users/andre/OneDrive/Systems/Lab06$ ./list science.uoit.ca http
IPv4 stream port: 80 address: 205.211.180.148
IPv4 stream port: 80 address: 205.211.180.149
(base) andre@DESKTOP-UM1B7BM:/mnt/c/Users/andre/OneDrive/Systems/Lab06$ ./list science.uoit.ca ssh
IPv4 stream port: 22 address: 205.211.180.149
IPv4 stream port: 22 address: 205.211.180.148
(base) andre@DESKTOP-UM1B7BM:/mnt/c/Users/andre/OneDrive/Systems/Lab06$ ./list science.uoit.ca echo
IPv4 stream port: 7 address: 205.211.180.148
IPv4 datagram port: 7 address: 205.211.180.148
IPv4 stream port: 7 address: 205.211.180.149
IPv4 datagram port: 7 address: 205.211.180.149
(base) andre@DESKTOP-UM1B7BM:/mnt/c/Users/andre/OneDrive/Systems/Lab06$
```

```c
/*****************************************
 *
 *                  list.c
 *
 *  Walk the result returned by getaddrinfo
 *  and print the information that we find.
 *****************************************/

#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>

int main(int argc, char *argv[]) {
    char *host;
    char *service;
    struct addrinfo hints;
    struct addrinfo *addr;
    struct addrinfo *save;
    struct sockaddr_in *addrinfo;
    int rc;
```

```
    char address[512];

    /*
     *  check for proper usage
     */
    if(argc != 3) {
        printf("Usage: list hostname service\n");
        exit(1);
    }

    host = argv[1];
    service = argv[2];

    /*
     *  request all protocol families
     */
    memset(&hints, 0, sizeof(hints));
    hints.ai_family = AF_UNSPEC;

    rc = getaddrinfo(host, service, &hints, &addr);
    if(rc != 0) {
        printf("getaddrinfo failed: %s\n", gai_strerror(rc));
        exit(1);
    }

    /*
     *  save the result so we can free it later
     */
    save = addr;

    while(addr != NULL) {
        if(addr->ai_family == AF_INET)
            printf("IPv4 ");
        else if(addr->ai_family == AF_INET6)
            printf("IPv6 ");
        else
            printf("Unknown family ");

        if(addr->ai_socktype == SOCK_STREAM)
            printf("stream ");
```

```
        else if(addr->ai_socktype == SOCK_DGRAM)
            printf("datagram ");
        else
            printf("unknown socket type ");

        addrinfo = (struct sockaddr_in *) addr->ai_addr;
        printf("port: %d ", ntohs(addrinfo->sin_port));
        /*
         *  convert the IP address to human readable form
         */
        inet_ntop(addr->ai_family, (void*) &addrinfo->sin_addr,
            address, sizeof(address));
        printf("address: %s ",address);
        printf("\n");
        addr = addr->ai_next;
    }

    freeaddrinfo(save);

    exit(0);
}
```