CLIENT:

```
#include <stdio.h>
#include <string.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define PORT 7228

void printOption(time_t cur_time, int opt);
int main(int argc, char **argv){
        time_t cur_time;
        struct sockaddr_in serv_addr, cli_addr;
        int sockfd, n, addrlen, opt, cont;

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);
        n = atoi(argv[1]);

        if(sockfd < 0){
                perror("Error in opening socket.\n");
                exit(1);
        }

        bzero(&serv_addr, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = INADDR_ANY;
        serv_addr.sin_port = htons(PORT);

        addrlen = sizeof(serv_addr);

        while(1){
                printf("\nRequest to Server:\n\t1 - Date\n\t2 - Day\n\t3 -
Month\n\t4 - Year\n\t5 - Time\n\t6 - Toronto Time\nOption -> ");
                scanf("%d", &opt);
                sendto(sockfd, &n, sizeof(n), 0, (struct
sockaddr*)&serv_addr, sizeof(serv_addr));
                recvfrom(sockfd, &cur_time, sizeof(cur_time), 0, (struct
sockaddr*)&serv_addr, &addrlen);
                printOption(cur_time, opt);
                printf("\nDo you want to continue? (0/1) -> ");
                scanf("%d", &cont);

                if(cont == 0){
                        break;
                }

        }

        close(sockfd); //close client sockfd once requests are over.

        return 0;
}

void printOption(time_t cur_time, int opt){
        struct tm *temp;
        time_t toronto_time;
        char time_buffer[1000];
```

```
        temp = localtime(&cur_time);

        switch(opt){
                case 1:
                                strftime(time_buffer, sizeof(time_buffer),
"%x", temp);
                                printf("Date: %s\n", time_buffer);
                                break;
                case 2:
                                strftime(time_buffer, sizeof(time_buffer),
"%A", temp);
                                printf("Day of Week\t:\t%s\n", time_buffer);
                                strftime(time_buffer, sizeof(time_buffer),
"%d", temp);
                                printf("Day of Month\t:\t%s\n",
time_buffer);
                                bzero(&time_buffer, sizeof(time_buffer));
                                strftime(time_buffer, sizeof(time_buffer),
"%j", temp);
                                printf("Day of Year\t:\t%s\n", time_buffer);
                                break;
                case 3:
                                strftime(time_buffer, sizeof(time_buffer),
"%B", temp);
                                printf("Month\t:\t%s\n", time_buffer);
                                break;
                case 4:
                                printf("Year\t:\t%d\n", (temp->tm_year +
1900)); //tm_year stores years elapsed since Unix epoch
                                break;
                case 5:
                                strftime(time_buffer, sizeof(time_buffer),
"%I:%M%p", temp);
                                printf("Time\t:\t%s\n", time_buffer);
                                break;
                case 6:
                                strftime(time_buffer, sizeof(time_buffer),
"%c", temp);
                                printf("Local Time\t:\t%s\n", time_buffer);
                                bzero(&time_buffer, sizeof(time_buffer));
                                temp = gmtime(&cur_time);
                                temp->tm_hour -= 5;

                                toronto_time = mktime(temp);
                                temp = localtime(&toronto_time);
                                strftime(time_buffer, sizeof(time_buffer),
"%c", temp);
                                printf("Toronto Time\t:\t%s\n",
time_buffer);
                                break;
                default:
                                printf("\n\tInvalid option.\n");
                                break;
        }

}
```

SERVER:

```
#include <stdio.h>
#include <string.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <sys / socket.h>
#include <netinet / in.h>

#define PORT 7228

int main (int argc, char ** argv) {
        time_t cur_time;
        struct sockaddr_in serv_addr, cli_addr;
        int sockfd, n, addrlen;

        sockfd = socket (AF_INET, SOCK_DGRAM, 0);

        if (sockfd <0) {
                perror ("Can't open Socket \ n");
                exit (1);
        }

        bzero (& serv_addr, sizeof (serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = INADDR_ANY;
        serv_addr.sin_port = htons (PORT);

        if (bind (sockfd, (struct sockaddr *) & serv_addr, sizeof
(serv_addr)) <0) {
                // Binding the socket to the port with serv_addr
                perror ("Binding Error! \ n");
                exit (1);
        }

        addrlen = sizeof (cli_addr);

        while (1) {// server is always up
                recvfrom (sockfd, & n, sizeof (n), 0, (struct sockaddr *) &
cli_addr, & addrlen);
                cur_time = time (NULL);
                printf ("\ nClient% d is requesting through port% d. \ n",
n, PORT);
                sendto (sockfd, & cur_time, sizeof (cur_time), 0, (struct
sockaddr *) & cli_addr, addrlen);
                printf ("Sent current time:% s", ctime (& cur_time));
        }

        return 0;
}
```