Nom : Num Etudiant :
Prenom :

Exercice 2 :

#	Num Seq	Num Ack	URG	ACK	PSH	RST	SYN	FIN	Taille Données
1	2499	-					X		0
2	3523	***2500***		X			X		0
3	***2500***	***3524***		X					0
4	***2500***	***3524***							200
5	***2700***	***3524***							200
6	***3524***	***2700***		***X***					100
7	***3624***	***2900***		***X***					***100***
8	***2900***	***3624***		***X***					***200***
9	***3100***	***3724***		***X***					***200***
10	***3724***	***2900***		***X***					***100***
11	***3824***	***3300***		***X***					***100***
12	***3300***	***3724***		***X***					***200***
13	***3924***	***3300***							***100***
14	***3500***	***3724***		***X***					***200***
15	***3724***	***3300***		***X***					***100***
16	***3700***	***3824***		***X***					***200***
17	***3824***	***3300***		***X***					***100***
18	***3300***	***3924***		***X***					***200***
19	***3924***	***3300***		***X***					***100***
20	***3500***	***4024***		***X***					***200***
21	***4024***	***3700***		***X***					***100***
22	***3700***	***4124***		***X***					***200***
23	***4124***	***3900***		***X***					***100***
24	***3900***	***4224***		***X***				X	0
25	***4224***	***3901***		X					100
26	***4324***	***-***						X	0
27	***-***	***4325***		X					***0***

ANNEXE B

```
 1 /* clientu.c (client UDP) */
 2
 3 #include <stdio.h>
 4 #include <errno.h>
 5 #include <netinet/in.h>
 6 #include <string.h>
 7
 8 short sport = 0;
 9 int sock = 0; /* socket de communication */
10
11
12 int main(int argc, char** argv) {
13   struct sockaddr_in moi; /* SAP du client */
14   struct sockaddr_in serveur; /* SAP du serveur */
15
16   int ret,len,p,q,i;
17
18   int serveur_len = sizeof(serveur);
19   char buf_read[101], buf_write[101];
20
21   if (argc != 5) {
22       fprintf(stderr,"usage: %s host sport Q P\n",argv[0]);
23       exit(1);
24   }
25   sport = atoi(argv[2]);
26   q = atoi(argv[3]);
27   p = atoi(argv[4]);
28
29   if ((sock = socket(AF_INET,SOCK_DGRAM,0)) == -1) {
30       fprintf(stderr,"%s: socket %s\n",argv[0],strerror(errno));
31       exit(1);
32   }
33
34   len = sizeof(moi);
35   getsockname(sock,(struct sockaddr *)&moi,&len);
36   serveur.sin_family = AF_INET;
37   serveur.sin_port = htons(sport);
38   inet_aton(argv[1],(struct in_addr *)&serveur.sin_addr);
39
40   sprintf(buf_write,"%d %d",q,p);
41
42   while (strlen(buf_write) < 100)
43       strcat(buf_write," ");
44
45   ret = sendto(sock,buf_write,strlen(buf_write)+1,0,(struct sockaddr
*)&serveur,sizeof(serveur));
46
47   if (ret <= 0) {
48       printf("%s: erreur dans sendto (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
49       return -1;
50   }
51
52   len = sizeof(moi);
53   getsockname(sock,(struct sockaddr *)&moi,&len);
54   printf("le client recoit : ");
```

```
 55   ret = recvfrom(sock,buf_read,101,0,(struct sockaddr
*)&serveur,&serveur_len);
 56   if (ret <= 0) {
 57       printf("%s: erreur dans recvfrom (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
 58       return -1;
 59   }
 60
 61   printf("%s\n",buf_read);
 62   for (i = 0; i < q;i++) {
 63       getsockname(sock,(struct sockaddr *)&moi,&len);
 64       printf("le client recoit ");
 65       ret = recvfrom(sock,buf_read,101,0, (struct sockaddr
*)&serveur,&serveur_len);
 66       if (ret <= 0) {
 67           printf("%s: erreur dans recvfrom (num=%d, mess=%s)\n",
argv[0],ret,strerror(errno));
 68           return -1;
 69       }
 70       printf("%s\n",buf_read);
 71   }
 72
 73   getsockname(sock,(struct sockaddr *)&moi,&len);
 74   printf("le client recoit : ");
 75   ret = recvfrom(sock,buf_read,101,0,(struct sockaddr
*)&serveur,&serveur_len);
 76   if (ret <= 0) {
 77       printf("%s: erreur dans recvfrom (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
 78       return -1;
 79   }
 80   printf("%s\n",buf_read);
 81   return 0;
 82 }
```

```
  1 /* serveuru.c (serveur UDP) */
  2
  3 #include <stdio.h>
  4 #include <errno.h>
  5 #include <string.h>
  6 #include <netinet/in.h>
  7
  8 short port = 0;
  9 int sock = 0; /* socket de communication */
 10 int nb_reponse = 0;
 11
 12 int main(int argc, char** argv) {
 13     int ret;
 14     struct sockaddr_in serveur; /* SAP du serveur */
 15
 16     if (argc != 2) {
 17         fprintf(stderr,"usage: %s port\n",argv[0]);
 18         exit(1);
 19     }
 20     port = atoi(argv[1]);
 21
```

```
 22     if ((sock = socket(AF_INET,SOCK_DGRAM,0)) == -1) {
 23         fprintf(stderr,"%s: socket %s\n",argv[0],strerror(errno));
 24         exit(1);
 25     }
 26
 27     serveur.sin_family = AF_INET;
 28     serveur.sin_port = htons(port);
 29     serveur.sin_addr.s_addr = INADDR_ANY;
 30     if (bind(sock,(struct sockaddr *)&serveur,sizeof(serveur)) < 0) {
 31         fprintf(stderr,"%s: bind %s\n", argv[0],strerror(errno));
 32         exit(1);
 33     }
 34
 35     while (1) {
 36         struct sockaddr_in client; /* SAP du client */
 37         int client_len = sizeof(client);
 38         char buf_read[101], buf_write[101];
 39         char chaine[10];
 40
 41         int i,j,p,q;
 42
 43         ret = recvfrom(sock,buf_read,100,0, (struct sockaddr
*)&client,&client_len);
 44
 45         if (ret <= 0) {
 46             printf("%s: recvfrom=%d: %s\n", argv[0],ret,strerror(errno));
 47             return -1;
 48         }
 49
 50         chaine[0]='\0';
 51         for (i = 0; buf_read[i] != ' '; i++)
 52             chaine[i] = buf_read[i];
 53
 54         chaine[i]='\0';
 55         q = atoi(chaine);
 56         chaine[0]='\0';
 57
 58         for (i++, j = 0; buf_read[i] != ' '; i++, j++)
 59          chaine[j] = buf_read[i];
 60         chaine[j]='\0';
 61
 62         p = atoi(chaine);
 63
 64         printf("serveur a recu p=%d, q=%d\n",p,q);
 65         strcpy(buf_write,"OK");
 66         while (strlen(buf_write) < 100)
 67             strcat(buf_write," ");
 68
 69         ret = sendto(sock,buf_write,101,0, (struct sockaddr
*)&client,client_len);
 70         if (ret <= 0) {
 71               printf("%s: sendto=%d: %s\n", argv[0],ret,strerror(errno));
 72             return -1;
 73         }
 74
 75         for (i = 0; i < q;) {
 76             sleep(p);
 77             sprintf(buf_write,"S %d",++i);
```

```
 78             while (strlen(buf_write) < 100)
 79                 strcat(buf_write," ");
 80             ret = sendto(sock,buf_write,101,0, (struct sockaddr
*)&client,client_len);
 81             if (ret <= 0) {
 82                 printf("%s: sendto=%d: %s\n",argv[0],ret,strerror(errno));
 83                 return -1;
 84             }
 85         }
 86
 87         strcpy(buf_write,"FIN");
 88         while (strlen(buf_write) < 100)
 89             strcat(buf_write," ");
 90         ret = sendto(sock,buf_write,101,0,(struct sockaddr
*)&client,client_len);
 91         if (ret <= 0) {
 92             printf("%s: sendto=%d: %s\n", argv[0],ret,strerror(errno));
 93             return -1;
 94         }
 95     }
 96     return 0;
 97 }
```

```
  1 /* serveurt.c (serveur TCP) */
  2
  3 #include <stdio.h>
  4 #include <errno.h>
  5 #include <netinet/in.h>
  6 #include <string.h>
  7
  8 short port = 0;
  9 int sock = 0; /* socket de communication */
 10
 11 int main(int argc, char** argv) {
 12     struct sockaddr_in serveur; /* SAP du serveur */
 13
 14     if (argc != 2) {
 15         fprintf(stderr,"usage: %s port\n",argv[0]);
 16         exit(1);
 17     }
 18     port = atoi(argv[1]);
 19     if ((sock = socket(AF_INET,SOCK_STREAM,0)) == -1) {
 20         fprintf(stderr,"%s: socket %s\n",argv[0],strerror(errno));
 21         exit(1);
 22     }
 23     serveur.sin_family = AF_INET;
 24     serveur.sin_port = htons(port);
 25     serveur.sin_addr.s_addr = INADDR_ANY;
 26     if (bind(sock,(struct sockaddr *)&serveur,sizeof(serveur)) < 0) {
 27         fprintf(stderr,"%s: bind %s\n",argv[0],strerror(errno));
 28         exit(1);
 29     }
 30     if (listen(sock,5) != 0) {
 31         fprintf(stderr,"%s: listen %s\n",argv[0],strerror(errno));
 32         exit(1);
 33     }
```

```
34      while (1) {
35          struct sockaddr_in client; /* SAP du client */
36          int len = sizeof(client);
37          int sock_pipe; /* socket de dialogue */
38          int ret,p,q,i;
39
40          sock_pipe = accept(sock,(struct sockaddr *)&client,&len);
41          ret = read(sock_pipe,&q,sizeof(q));
42          if (ret <= 0) {
43              printf("%s: read=%d: %s\n",argv[0], ret, strerror(errno));
44              return -1;
45          }
46          ret = read(sock_pipe,&p,sizeof(p));
47          if (ret <= 0) {
48              printf("%s: read=%d: %s\n",argv[0], ret, strerror(errno));
49              return -1;
50          }
51          printf("serveur a recu p=%d, q=%d\n",p,q);
52          ret = write(sock_pipe,"OK",2);
53          if (ret <= 0) {
54              printf("%s: write=%d: %s\n", argv[0], ret, strerror(errno));
55              return -1;
56          }
57          for (i = 0 ; i < q;) {
58              sleep(p);
59              ret = write(sock_pipe,"S",1);
60              if (ret <= 0) {
61                  printf("%s: write=%d: %s\n",argv[0], ret, strerror(errno));
62                  return -1;
63              }
64              i++;
65              ret = write(sock_pipe,&i,sizeof(i));
66              if (ret <= 0) {
67                  printf("%s: write=%d: %s\n",argv[0], ret, strerror(errno));
68                  return -1;
69              }
70          }
71          ret = write(sock_pipe,"END",3);
72          if (ret <= 0) {
73              printf("%s: write=%d: %s\n",argv[0], ret, strerror(errno));
74              return -1;
75          }
76          close(sock_pipe);
77      }
78      return 0;
79  }
80
```

```
1  /* clientt.c (client TCP) */
2
3  #include <stdio.h>
4  #include <errno.h>
5  #include <netinet/in.h>
6  #include <string.h>
7
8  short sport = 0;
```

```
 9 int sock = 0; /* socket de communication */
10
11 int main(int argc, char** argv) {
12     struct sockaddr_in moi; /* SAP du client */
13     struct sockaddr_in serveur; /* SAP du serveur */
14     int ret,len,p,q,i,num;
15     char buf_read[4];
16
17     if (argc != 5) {
18         fprintf(stderr,"usage: %s serveur port Q P\n",argv[0]);
19         exit(1);
20     }
21     sport = atoi(argv[2]);
22     q = atoi(argv[3]);
23     p = atoi(argv[4]);
24     if ((sock = socket(AF_INET,SOCK_STREAM,0)) == -1) {
25         fprintf(stderr,"%s: socket %s\n",argv[0],strerror(errno));
26         exit(1);
27     }
28     serveur.sin_family = AF_INET;
29     serveur.sin_port = htons(sport);
30     inet_aton(argv[1],(struct in_addr *)&serveur.sin_addr);
31     if (connect(sock,(struct sockaddr *)&serveur,sizeof(serveur)) < 0) {
32         fprintf(stderr,"%s: connect %s\n",argv[0],strerror(errno));
33         perror("bind");
34         exit(1);
35     }
36     len = sizeof(moi);
37     getsockname(sock,(struct sockaddr *)&moi,&len);
38     ret = write(sock,&q,sizeof(q));
39     if (ret < 2) {
40         printf("%s: erreur dans write (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
41         return -1;
42     }
43     ret = write(sock,&p,sizeof(p));
44     if (ret < 2) {
45         printf("%s: erreur dans write (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
46         return -1;
47     }
48     ret = read(sock,buf_read,2);
49     if (ret <= 0) {
50         printf("%s: erreur dans read (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
51         return -1;
52     }
53     printf("Le client recoit %s\n",buf_read);
54     for (i = 0; i < q; i++) {
55         ret = read(sock,buf_read,1);
56         if (ret <= 0) {
57             printf("%s: erreur dans read (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
58             return -1;
59         }
60         buf_read[1] = '\0';
61         ret = read(sock,&num,sizeof(num));
62         if (ret <= 0) {
```

```
 63             printf("%s: erreur dans read (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
 64             return -1;
 65         }
 66         printf("Le client recoit %s%d\n",buf_read,num);
 67     }
 68     ret = read(sock,buf_read,3);
 69     if (ret <= 0) {
 70         printf("%s: erreur dans read (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
 71         return -1;
 72     }
 73     buf_read[3] = '\0';
 74     printf("Le client recoit %s\n",buf_read);
 75     close(sock);
 76     return 0;
 77 }
```