CSE 344 FINAL PROJECT REPORT
SEFA NADİR YILDIZ
131044031

server.c

```
if (argc != 4) {
      fprintf(stderr, "Usage: executableFile connectionPort providersFile logFile\n");
      exit(0);
}
```

Command line parametrelerini kontrol ettikten sonra öncelikli olarak data.dat dosyasından provider bilgilerini elde etmek için bu dosyayı okuyup parse ettim.

calculate_row_number fonksiyonunda kaç adet provider olduğunu bularak

```
typedef struct provider_type {
   char name[SIZE];
   int performance;
   int price;
   int duration;
   int queue[2];
} provider;
```

provider struct tipindeki array değişkenimin alanını malloc ile aldım.

```
for (i = 0; i < size; ++i) {
pthread_create(&(client_teachers[i]), NULL, &processingRequests (void*)&providers[i]);
if (thread_error != 0)
fprintf(stderr, "Thread can't be created :[%s]", strerror(thread_error));
}
```

Yukarıda her provider için thread create edip pool'da beklemelerini sağladım.

Socket yapılarımı aşağıdaki gibi kurdum.

```
   if (socket_desc == -1) {
      fprintf(stderr, "Could not create socket\n");
      exit(0);
   }
   true = 1;
   if (setsockopt(socket_desc, SOL_SOCKET, SO_REUSEADDR, &true, sizeof (int)) == -1) {
      fprintf(stderr, "Set sock opt error\n");
      exit(0);
   }
   server.sin_family = AF_INET;
   server.sin_addr.s_addr = INADDR_ANY;
   server.sin_port = htons(atoi(argv[1]));

   if (bind(socket_desc, (struct sockaddr *) &server, sizeof (server)) < 0) {
      fprintf(stderr, "bind failed\n");
      exit(0);
   }
```

Bu kodda yeni client gelip gelmediğini kontrol ettim ve her yeni client için bir thread oluşturdum.

```
   while ((client_sock = accept(socket_desc, (struct sockaddr *) &client, (socklen_t*) & c))) {
      puts("Connection accepted");
      sleep(1);
      pthread_t sniffer_thread;
      new_sock = malloc(1);
      *new_sock = client_sock;

      if (pthread_create(&sniffer_thread, NULL, connection_handler, (void*) new_sock) < 0) {
         perror("could not create thread");
         return 1;
      }
      pthread_join(sniffer_thread, NULL);
      puts("Handler assigned");
   }
```

void *connection_handler(void *socket_desc) bu thread fonksıyonun clienttan gelecek olan mesajları **if ((read_size = recv(sock, client_message, SIZE, 0)) > 0)** kontrol ettim.

Client'ın aradığı özelliklere pars işlemi yapıp ilgili provide ı bulduktan sonra **void* processingRequests(void *arg)** thread fonksıyonunda kontroller yaptım.

```
   while (1) {
      pthread_mutex_lock(&lock_provider);
      if (strcmp((((provider*) arg)->name), finded_provider_name) == 0) {
         …….
}
```

Sonuçları hesaplayıp ekrana bastırdıktan sonra clienta mesaj olarak su sekılde gonderdım

```
        fprintf(stderr, "%s's task completed by %s in %.2f seconds, cos(%d)=%.2f, cost is %d total
time spent %.2f seconds.\n",
                one_client.name,
                ((provider*) arg)->name,
                time,
                one_client.homework,
                result,
                ((provider*) arg)->price,
                time);
          strcat(finded_provider_name, " ");
          sprintf(sent_message, "%s's task completed by %s in %.2f seconds, cos(%d)=%.2f, cost is
%d total time spent %.2f seconds.\n",
                one_client.name,
                ((provider*) arg)->name,
                time,
                one_client.homework,
                result,
                ((provider*) arg)->price,
                time);
```

client.c

por yapısını olusturdum.

```
 if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        fprintf(stderr, "Socket didn't create\n");
        exit(0);
    }
    memset(&serv_addr, '0', sizeof (serv_addr));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(atoi(argv[5]));

    if (inet_pton(AF_INET, argv[4], &serv_addr.sin_addr) <= 0) {
        fprintf(stderr, "Invalid address\n");
        exit(0);
    }
    if (connect(sock, (struct sockaddr *) &serv_addr, sizeof (serv_addr)) < 0) {
        fprintf(stderr, "Connection failed \n");
        exit(0);
    }
```

Gonderılecek mesajı bırlestırerek server'a gonderdım

```
    strcpy(request_message, argv[1]);
    strcat(request_message, " ");
    strcat(request_message, argv[2]);
    strcat(request_message, " ");
    strcat(request_message, argv[3]);
    strcat(request_message, " ");
    sprintf(pid_array, "%d", getpid());
    strcat(request_message, pid_array);

    send(sock, request_message, strlen(request_message), 0);
```