```
#define QueueLimit ???

typedef struct {                        //Τροποποίηση του τύπου των στοιχείων της ουράς
  ???                                   // για να καταχωρούνται πελάτες
} QueueElementType;

??? ReadCustomer(???);                                  //Συνάρτηση για το (1)
QueueType TimesInQueue(QueueType *Queue);               //Συνάρτηση για το (2)
??? TraverseQ(???, ???);                                //Συνάρτηση για το (3)

int main(){
        Δήλωσε την ουρά WaitingQueue            //Ουρά αναμονής
        Δήλωσε την ουρά ServiceQueue            //Ουρά εξυπηρετηθέντων


        Δημιουργία της ουράς αναμονής
        Με επαναληπτική δομή (βρόχος for) για το μέγιστο πλήθος πελατών {
                Διάβασε τον επόμενο πελάτη (ReadCustomer(???))
        }

        Εμφάνισε την ουρά αναμονής (TraverseQ(???, ???))

        Προσομοίωσε την εξυπηρέτηση των πελατών και επέστρεψε την ουρά
        εξυπηρετηθέντων

        Εμφάνισε την ουρά αναμονής (TraverseQ(???, ???))
        Εμφάνισε την ουρά εξυπηρετηθέντων (TraverseQ(???, ???))
        return 0;
}




void ReadCustomer(???){
        Δήλωση μεταβλητών
        ??? tmpCustomer;

        Διάβασε την ώρα άφιξης και τον χρόνο αναμονής  //οι δύο αριθμοί δίνονται με
                                                        //κόμμα μεταξύ τους
        Αρχικοποίησε ώρα έναρξης και ώρα λήξης
        Εισήγαγε τον πελάτη στην ουρά αναμονής (AddQ(???,???))
```

```
}

QueueType TimesInQueue(QueueType *Queue) {     //Παράμετρος η ουρά αναμονής
        Δήλωση μεταβλητών
        int earliestStart =0;              //χρόνος έναρξης εξυπηρέτησης
        int currentStart=0;                //ώρα έναρξης εξυπηρέτησης για τρέχοντα πελάτη
        int currentEnd=0;                  //ώρα λήξης εξυπηρέτησης για τρέχοντα πελάτη


        ???     QueueServiced;             //Ουρά εξυπηρετηθέντων
        ???     CurrentCust;

        Δημιουργία ουράς εξυπηρέτησης

        Όσο η ουρά αναμονής δεν είναι κενή{
                Αφαίρεσε τον πελάτη από την αρχή της ουράς αναμονής (RemoveQ(???,???))

                Αν η ώρα άφιξης του πελάτη είναι μετά τον χρόνο έναρξης
                εξυπηρέτησης
                        Θέσε την τρέχουσα ώρα έναρξης ίση με την ώρα άφιξης πελάτη
                Αλλιώς
                        Θέσε την τρέχουσα ώρα έναρξης ίση με τον χρόνο έναρξης
                εξυπηρέτησης

                Ενημέρωσε την  ώρα λήξης εξυπηρέτησης για τον τρέχοντα πελάτη
                (=τρέχουσα ώρα έναρξης +χρόνος παραμονής)

                Ενημέρωσε τον χρόνο έναρξης εξυπηρέτησης για τον επόμενο πελάτη
                (=χρόνος λήξης τρέχοντα πελάτη)

                //Ενημέρωσε τον τρέχοντα πελάτη με τα νέα του στοιχεία
                CurrentCust.start=???
                CurrentCust.end=???

                Εισήγαγε τον τρέχοντα πελάτη στην ουρά εξυπηρετηθέντων (AddQ(???,???))
        }
        Επέστρεψε την ουρά εξυπηρετηθέντων
}
```

```
void TraverseQ(???, QueueType Queue){
        int current;
        int i=???;                          //βοηθητικός μετρητής πελατών

        Αν η ουρά δεν είναι κενή{
                Εμφάνισε την ονομασία της ουράς

                current = Queue.Front;
                printf("Client  \t\tStart\tEnd\tArrival\tStay\n");

                while (current != Queue.Rear) {
                Εμφάνισε τον πελάτη i, και όλα του τα στοιχεία (ώρα έναρξης εξυπηρέτησης,
                ώρα, λήξης εξυπηρέτησης, ώρα άφιξης, χρόνος παραμονής)

                current = (current + 1) % QueueLimit;
        }
        Αλλιώς
                Εμφάνισε ότι η ουρά είναι κενή       //να εμφανίζεται η ονομασία της ουράς
}
```