Si deve realizzare la simulazione di un vigile urbano virtuale che regoli il traffico ad un quadrivio. Sono definiti tre differenti tipi di quadrivio: all'italiana, all'americana e a rotatoria. Ogni gruppo deve realizzare almeno uno dei tre tipi di quadrivio proposti.

All'italiana: delle quattro strade che si incrociano almeno una (ma anche due) hanno un segnale per dare la precedenza a tutti, coloro che arrivano dalle rimanenti strade danno la precedenza a chi arriva dalla loro destra.

All'americana: nessuno è specificamente obbligato a dare la precedenza ad altri, passa per primo chi è arrivato per primo all'incrocio.

A rotatoria: le strade sboccano su una struttura circolare all'interno della quale sono già presenti altre macchine che ruotano in fila indiana. Chi è in testa ad ogni strada aspetta che gli sia passata davanti l'ultima macchina del cerchio parziale e si accoda. Le macchine lasciano la struttura circolare ed escono quando "vedono" la strada che devono impegnare. La rotatoria può essere generalizzata a n strade (cioè non necessariamente 4).

Non è necessario implementare più di uno dei tre i tipi di incrocio. Se volete farlo, bene, se lo fate ma non ne siete sicuri tenete presente che ogni cosa che presenterete potrà essere usata contro di voi. Sarà meglio fare poco e farlo bene piuttosto che molto e male.

Aspetti implementativi:
ogni strada che confluisce nel quadrivio deve essere simulata con la coda delle macchine che la occupa, nel caso dell'incrocio all'americana i nodi nella coda devono contenere anche dati sull'ordine di arrivo all'incrocio.
Durante le precedenti esercitazioni, sono state sperimentate code statiche con array, code simulate con doppio stack sia statiche che dinamiche, code dinamiche vere e proprie. Dovete usare quanto più codice precedentemente prodotto possibile.
Avete due possibilità: o scrivete delle librerie con gli stessi nomi delle chiamate che implementano due gestioni di coda differenti e dimostrate che il vostro main funziona allo stesso modo con entrambe le librerie senza cambiare nemmeno una linea di codice (soluzione preferita),
oppure implementate le code in modi diversi per ogni "strada" tenendo insieme le librerie, ovviamente usando diversi nomi per le chiamate dei due tipi.
Uno sguardo attento deve essere rivolto al problema della architettura del progetto. Dovete specificamente separare le funzioni del main da quelle delle procedure che sono scritte espressamente per risolvere il problema risolto e dovete inoltre ben differenziare il ruolo di queste procedure rispetto alle funzioni di libreria che devono essere intese invece come strumenti di uso generale e per le quali deve essere garantita una totale portabilità verso altri programmi.

Le code devono essere riempite con dati casuali all'inizio e non a mano al runtime, dopo lo svuotamento dell'incrocio il programma termina, è previsto l'arrivo di sole ulteriori 4 macchine nel corso del programma solo per mostrare che sapete fare un "enqueue" al volo. Nel caso dell'incrocio a rotatoria deve essere implementata una ulteriore struttura dati che simuli la rotatoria, serve una lista circolare, o è sufficiente un'altra coda?
A voi la risposta.

Altre cose lasciate alla vostra scelta:
come far vedere all'utente quello che succede all'incrocio, ma non usate la grafica, fare i cartoni animati non aumenterà il voto che vi daremo.

Decidete voi se è importante o no conservare nelle strutture dati anche in quale strada vogliono andare le macchine all'incrocio, comunque una volta uscite dalle rispettive code le macchine se ne vanno e non vogliamo saperne più nulla (tranne ovviamente che nel caso della rotatoria).

L'ultima data utile per la consegna è il 9.1.2006, presto pubblicheremo sul sito le modalità di consegna.

Il programma deve essere consegnato in codice sorgente e deve fare uso di file header che contengano le dichiarazioni prototipali e l'implementazione delle procedure (non compilate separatamente in .o), la sua compilazione non deve porre problemi al valutatore utilizzando il dev c. Insieme al codice sorgente devono essere consegnati anche la relativa documentazione e una versione eseguibile del programma stesso. Pur essendo il lavoro valutato nel suo complesso con un unico voto di gruppo, richiediamo che ogni per singola procedura nel codice sia indicato il nome del membro del gruppo che se ne dichiara responsabile.

Su quali basi valuteremo il vostro lavoro:
Il programma deve essere fatto rispettando criteri di leggibilità del codice, di semplicità di gestione e di efficacia dell'interfaccia con l'utente che lo usa, la documentazione di progetto deve essere esauriente e ben scritta. Le procedure devono essere di uso generale e facilmente esportabili verso altri tipi di problemi, la massima specificità di risoluzione del problema dovrebbe essere affidata al main che dovrebbe limitarsi a chiamare una serie di procedure generiche che realizzino compiti quanto più standardizzati possibile. Il voto finale sarà espresso in 15simi e sarà mediato con il voto di un secondo progetto che verrà proposto a gennaio a fine corso. Il voto finale in 15simi sarà sommato al voto in 15simi dello scritto che sarà fatto negli appelli di gennaio e febbraio. Tutto ciò vale solo per i frequentanti e solo per i due appelli immediatamente dopo il corso. Per i non frequentanti e per coloro che non consegneranno i due progetti nei tempi stabiliti verranno proposte differenti (e più severe) modalità di esame.