typedef struct {

 int idCompagniaAerea;

 int idAeroportoPartenza;

 int numeroPasseggeri;

 time\_t dataOraAtterraggio;

 int trasportoEccezionale; //1 normale, 0 eccezionale

 Aereo \*next;

} Aereo;

typedef struct {

 int idAeroporto;

 Aereo \*elenco;

 Aeroporto \*next;

} Aeroporto;

int main() {

 Aeroporto \*t = NULL;

 /\* altro codice \*/

 return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* FUNZIONE 1 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

la funzione restituisce:

-1 in caso di errore allocazione nuovo aereo

-2 in caso di errore allocazione nuovo aeroporto

1 in caso di operazione conclusa correttamente

\*/

int arrivoAereo(Aeroporto \*\*t, int idAeroportoArrivo, int idCompagniaAerea, int idAeroportoPartenza, int numeroPasseggeri, time\_t dataOraAtterraggio, int trasportoEccezionale) {

 Aereo \*nuovoAereo;

 Aeroporto \*tmpAeroporto;

 nuovoAereo = creaAereo(idCompagniaAerea, idAeroportoPartenza, numeroPasseggeri, dataOraAtterraggio, trasportoEccezionale);

 if (nuovoAereo == NULL) {

 return -1;

 }

 tmpAeroporto = trovaCreaAeroporto(t, idAeroportoArrivo);

 if (tmpAeroporto == NULL) {

 return -2;

 }

 inerisciAereoInLista(&tmpAeroporto->elenco, nuovoAereo);

 return 1;

}

void inerisciAereoInLista(Aereo \*\*t, Aereo \*nuovoAereo) {

 Aereo \*tmpAereo, \*precAereo;

 precAereo = NULL;

 tmpAereo = \*t;

 while (tmpAereo != NULL) {

 if (nuovoAereo->trasportoEccezionale < tmpAereo->trasportoEccezionale ||

 (nuovoAereo->trasportoEccezionale == tmpAereo->trasportoEccezionale &&

 nuovoAereo->dataOraAtterraggio < tmpAereo->dataOraAtterraggio)) {

 nuovoAereo->next = tmpAereo;

 if (precAereo == NULL) {

 \*t = nuovoAereo;

 }

 else {

 precAereo->next = nuovoAereo;

 }

 return;

 }

 else {

 precAereo = tmpAereo;

 tmpAereo = tmpAereo->next;

 }

 }

 if (\*t == NULL) {

 \*t = nuovoAereo;

 }

 else {

 precAereo->next = nuovoAereo;

 }

}

Aereo\* creaAereo(int idCompagniaAerea, int idAeroportoPartenza, int numeroPasseggeri, time\_t dataOraAtterraggio, int trasportoEccezionale) {

 Aereo \*nuovo;

 nuovo = (Aereo\*)malloc(sizeof(Aereo));

 if (nuovo == NULL) {

 return NULL;

 }

 nuovo->next = NULL;

 nuovo->idCompagniaAerea = idCompagniaAerea;

 nuovo->idAeroportoPartenza = idAeroportoPartenza;

 nuovo->numeroPasseggeri = numeroPasseggeri;

 nuovo->dataOraAtterraggio = dataOraAtterraggio;

 nuovo->trasportoEccezionale = trasportoEccezionale;

 return nuovo;

}

Aeroporto\* trovaCreaAeroporto(Aeroporto \*\*t, int idAeroporto) {

 Aeroporto\* tmp;

 tmp = cercaAeroporto(\*t, idAeroporto);

 if (tmp == NULL) {

 return creaAeroporto(idAeroporto);

 }

 return tmp;

}

Aeroporto\* cercaAeroporto(Aeroporto \*t, int idAeroporto) {

 Aeroporto\* tmp;

 tmp = t;

 while (tmp != NULL) {

 if (tmp->idAeroporto == idAeroporto) {

 return tmp;

 }

 tmp = tmp->next;

 }

 return NULL;

}

Aeroporto\* creaAeroporto(int idAeroporto) {

 Aeroporto \*nuovo;

 nuovo = (Aeroporto\*)malloc(sizeof(Aeroporto));

 if (nuovo == NULL) {

 return NULL;

 }

 nuovo->next = NULL;

 nuovo->elenco = NULL;

 nuovo->idAeroporto = idAeroporto;

 return nuovo;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* FUNZIONE 2 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

la funzione restituisce:

-1 se l'aeroporto non Ã¨ stato trovato

il numero di passeggeri diversamente

\*/

int numeroPasseggeriAeroporto(Aeroporto \*t, int idAeroporto) {

 Aeroporto\* tmp;

 Aereo\* tmpAereo;

 int conta = 0;

 tmp = cercaAeroporto(t, idAeroporto);

 if (tmp == NULL) {

 return -1;

 }

 tmpAereo = tmp->elenco;

 while (tmpAereo != NULL) {

 conta += tmpAereo->numeroPasseggeri;

 tmpAereo = tmpAereo->next;

 }

 return conta;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* FUNZIONE 3 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

la funzione restituisce:

il numero di passeggeri da un certo aeroporto di partenza

\*/

int numeroPasseggeriAeroportoPartenza(Aeroporto \*t, int idAeroportoPartenza) {

 Aeroporto\* tmpAeroporto;

 Aereo\* tmpAereo;

 int conta = 0;

 tmpAeroporto = t;

 while (tmpAeroporto != NULL) {

 tmpAereo = tmpAeroporto->elenco;

 while (tmpAereo != NULL) {

 if (tmpAereo->idAeroportoPartenza == idAeroportoPartenza) {

 conta += tmpAereo->numeroPasseggeri;

 }

 tmpAereo = tmpAereo->next;

 }

 tmpAeroporto = tmpAeroporto->next;

 }

 return conta;

}