# Come inviare lo svolgimento

- Creare archivio (zip o altro formato) con nome in questo formato:
- MATRICOLA_COGNOME_NOME_DATALEZIONE

- Esempio:
- IN0500123_ROSSI_PAOLO_20190218

- Intestazione email: deve contenere la stringa «[SISTEMI OPERATIVI]»

# Esercizio individuale 3° round

- Task A: Implementazione in C di una libreria che implementa un array dinamico, indice di tipo long >= 0; il valore associato ad ogni cella è generico (void *)
- L'implementazione utilizza una linked list (riutilizzare l'esempio "11linked_list")
- Va utilizzata la struttura di programma che trovate qui:

https://www.github.com/marcotessarotto/exOpSys/exRound3

Consegna entro 31/03 ore 24:00

# Task A: Libreria «ARRAY_LIST»

```
typedef struct ARRAY_ITEM_STRUCT {
   long id;   // chiave; id deve essere >= 0
   void * data; // valore
   struct ARRAY_ITEM_STRUCT * next;
} ARRAY_ITEM;
typedef struct ARRAY_LIST_STRUCT {
  ARRAY_ITEM * head;
  ARRAY_ITEM * tail;
  long size; // da tenere sempre aggiornato
} ARRAY_LIST;
```

# Task A: Libreria «ARRAY_LIST»

// fare check su id >= 0!!! Messaggio di errore in caso, e return NULL o -1 al caso, per segnalare errore

**ARRAY_LIST * create_new_array01();** // crea ed inizializza un nuovo array list

**void put_item02(ARRAY_LIST * array, long id, void * data);** // aggiunge all'array list un nuovo item, la cui chiave è id ed il valore è data; se id era già presente, il valore è sovrascritto dal nuovo

**void * get_item03(ARRAY_LIST * array, long id);** // cerca e resituisce il valore associato alla chiave id; restituisce NULL se non presente

# Task A: Libreria «ARRAY_LIST»

**void * remove_item04(ARRAY_LIST * array, long id);** // rimuove, se presente, l'item individuato dalla chiave id; restituisce il valore che era associato alla chiave id (se era presente), altrimenti restituisce NULL

**void destroy_array05(ARRAY_LIST * array, void (*free_fun)(void *));** // free dell'intero array list, per ogni value da distruggere chiama free_fun

**long find_value06(ARRAY_LIST * array, void * data, int (*value_cmp)(void *, void*));** // cerca il valore data nell'array e restituisce la chiave corrispondente; restituisce -1 se non trovata; value_cmp è il comparatore di valori

**void enumerate_values07(ARRAY_LIST * array, int (*fun_ptr)(long, void *));** // enumera tutto il contenuto dell'array, chiama fun_ptr per ogni item; fun_ptr restituisce 0 per fermare l'enumerazione

# Task B: deserializzazione di dati

dal National Bureau of Economic Research ( http://www.nber.org/data/city-place-distance-database.html )

City Distances Database

recuperare il seguente files:
1 - FILE sf12010placename.csv (PLACENAME)
http://www.nber.org/distance/2010/sf1/place/sf12010placename.csv

"state","statename","place","placename"
"01","Alabama","00100","Abanda CDP"
"01","Alabama","00124","Abbeville city"
...
max(state) = 72  // 7 bit

max(place) = 89760  // 17 bit
numero righe:  29.514

# Task B

- la "chiave" di questa tabella è data dalla coppia (state, place)

- state e place si possono combinare in un unico indice:
- int key = state << 17 | place; // "left shift" ed "OR numerico"

- ricostruzione di state e place da indice:

- int state = key >> 17; // "right shift"
  int place = key & 0x1FFFF;  // "AND numerico"

# Task B

- Leggere tutto il file PLACENAME ed immagazzinarlo in una struttura dati di tipo ARRAY_LIST
- Il dato statename, placename può essere immagazzinato usando il tipo dati: struct NAMES { char statename[N]; char placename[M]; } // N ed M li determinate a partire dal file
- Fare delle prove di interrogazione sull’array
- Liberare tutta la memoria

# Questo servirà in un round successivo…

2 - FILE sf12010placedistance25miles.csv (DISTANCES)
http://www.nber.org/distance/2010/sf1/place/sf12010placedistance25miles.csv.zip
"state1","place1","mi_to_place","state2","place2"
"01","00100",3.25667879291035,"01","79344"
"01","00100",6.40932722330285,"01","59016"
…

ogni riga: distanza in miglia da (state1, place1) a (state2, place2) con distanza <= 25 miglia

cioè: grazie a questa tabella, per ogni cittadina, conosciamo tutte le località vicine, a distanza <= 25 miglia

numero righe: 1.340.021