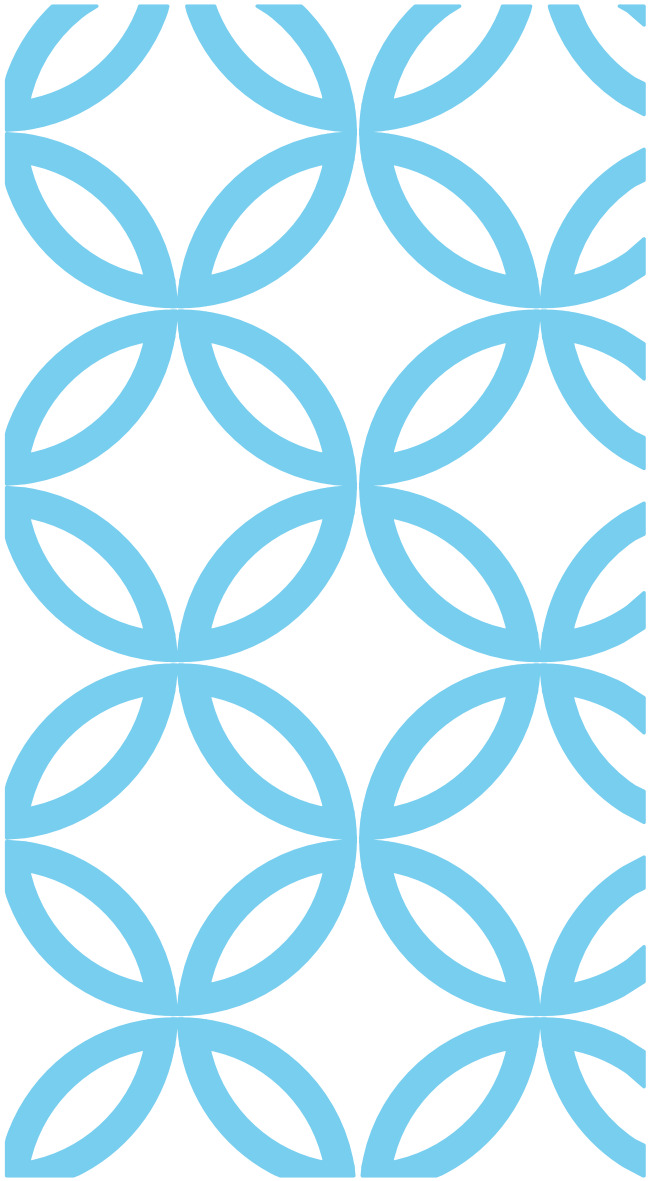

Corso di Programmazione e Strutture Dati

Docente di Laboratorio: Marco Romano
Email: marromano@unisa.it

# RICORSIONE

# ESERCITAZIONE SU OPERATORI RICORSIVI DI UNA LISTA

Sviluppare i seguenti operatori per l'ADT List che adoperino funzioni ricorsive per raggiungere l'obiettivo:

1. Stampa tutti gli elementi di una data lista
2. Conta il numero di volte che appare un dato Item in una data lista
3. Cerca un Item in una data lista
4. Deallocare tutti gli elementi di una data lista

# HEADER FILE

```
21  void printListRec (List list);
22  Item searchListRec (List list, Item item, int* pos);
23  int countItemListRec (List list, Item item);
24  void destroyListRec (List list);
```

# STAMPARE LA LISTA

```
226  void printNode (struct node *node)
227  {
228    if (node == NULL)
229      return;
230    outputItem (node -> item);
231    printNode (node -> next);
232  }
233
234  void printListRec (List list)
235  {
236    printNode (list -> head);
237  }
```

# HEADER FILE

```
21  void printListRec (List list);
22  Item searchListRec (List list, Item item, int* pos);
23  int countItemListRec (List list, Item item);
24  void destroyListRec (List list);
```

# CERCARE UN ITEM IN UNA LISTA

```
239  Item searchNode (struct node *node, Item item, int* pos)
240  {
241    if (node == NULL)
242    {
243      *pos = -1;
244      return NULL;
245    }
246    if (cmpItem (node -> item, item) == 0)
247      return node -> item;
248    else
249    {
250      ++*pos;
251      return searchNode (node -> next, item, pos);
252    }
253  }
254  Item searchListRec (List list, Item item, int* pos)
255  {
256    *pos = 0;
257    return searchNode (list -> head, item, pos);
258  }
```

# HEADER FILE

```
21  void printListRec (List list);
22  Item searchListRec (List list, Item item, int* pos);
23  int countItemListRec (List list, Item item);
24  void destroyListRec (List list);
```

# CONTARE IL NUMERO DI OCCORRENZE DI UN DATO ITEM

```
260  int countItemNode (struct node *node, Item item)
261  {
262    if(node == NULL)
263      return 0;
264    else{
265      if (cmpItem (node -> item, item) == 0)
266        return countItemNode(node->next, item) + 1;
267      else
268        return countItemNode(node->next, item);
269    }
270  }
271
272  int countItemListRec (List list, Item item)
273  {
274    return countItemNode(list->head, item);
275  }
```

# HEADER FILE

```
21  void printListRec (List list);
22  Item searchListRec (List list, Item item, int* pos);
23  int countItemListRec (List list, Item item);
24  void destroyListRec (List list);
```

# DEALLOCARE I NODI DI UNA LISTA

```
277  void destroyNode(struct node *node)
278  {
279    if(node){
280      destroyNode(node->next);
281      free(node);
282    }
283  }
284
285  void destroyListRec (List list)
286  {
287    destroyNode(list->head);
288    list->head = NULL;
289    list->size = 0;
290  }
```