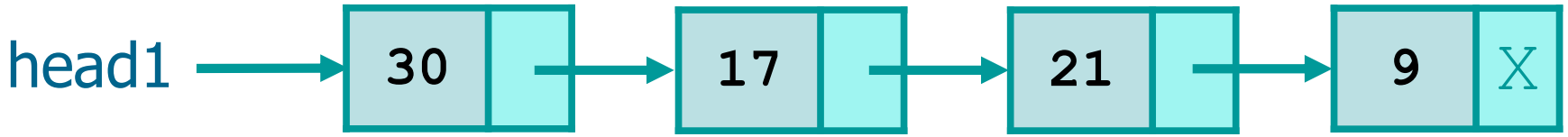
Linked Lists

Exercises

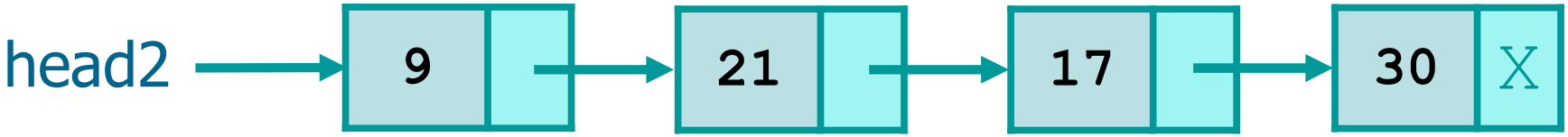
Dipartimento di Automatica e Informatica
Politecnico di Torino

List Inversion

- Given a simple linked list, invert all elements of such a list such that
  - The first element becomes the last one
  - The last element becomes the first one
  - Input list



  - Output list



Logic

- Strategy
  - Head1 points to the first list
    - The first list initially stores all elements
  - Head2 points to the second list
    - The second list in initially empty
  - While the first list is not empty
    - Extract element from the first list
      - Use head extraction
    - Insert element on the second list
      - Use head insertion

Implementation

```
typedef struct list_s list_t;
struct list_s {
  int key;
  ...
  list_t *next;
};
```

Type definition,

```
list_t *list_reverse (list_t *head1) {
  list_t *tmp1, head2;
  head2 = NULL;
  while (head1 != NULL) {
    tmp1 = head1->next;
    head1 = head1->next;
    tmp1->next = head2;
    head2 = tmp1;
  }
  return head2;
}
```

Extract from list 1

Insert into list 2

Implementation

```
typedef struct list_s list_t;
struct list_s {
  int key;
  ...
  list_t *next;
};
```

Type definition,

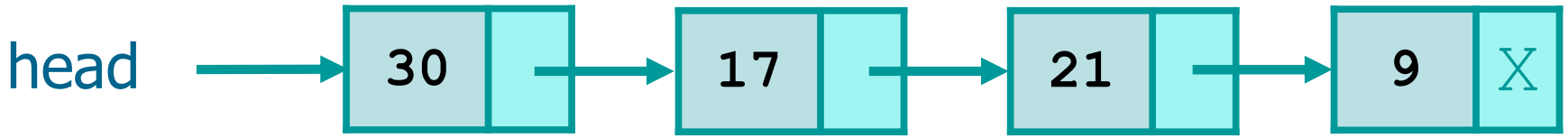
We can use pop and push

```
list_t *list_reverse (list_t *head1) {
  list_t head2 = NULL;
  int status;
  while (head1 != NULL) {
    head1 = pop (head1, &val, &status);
    if (status == SUCCESS)
      head2 = push (head2, val);
  }
  return head2;
}
```

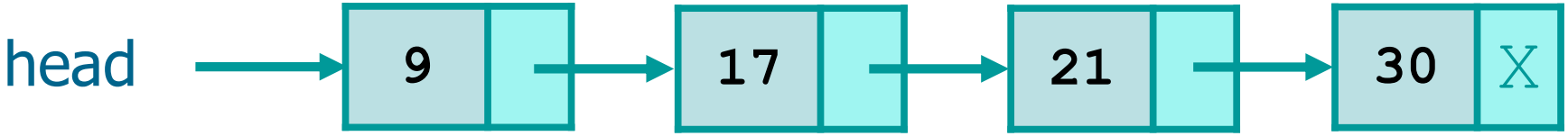
Modify the original pop and push appropriately

List Sort

- Given a simple linked list, sort all elements in ascending order
  - Input list



  - Output list



Logic

- Trivial implementation
  - Extract elements from the first list
    - Use head extraction (minimum cost)
  - Insert elements into the second list
    - Use insertion in-order position
  - Substitute first list with second list

Implementation

```
list_t *list_sort (list_t *head1) {
  list_t *tmp, *head2;
  head2 = NULL;
  while (head1 != NULL) {
    tmp = pop (&head1);
    head2 = insert (head2, tmp);
  }
  return head2;
}
```

Modify the original pop and insert appropriately
(to return and accept pointers to the elements, not the keys)