CORSO DI PROGRAMMAZIONE A-L
A.A. 2016-17

Dispensa 20

Laboratorio

Dott. Mirko Ravaioli
e-mail: mirko.ravaioli@unibo.it

20 Liste con priorità o lista ordinata

Le liste con priorità (o lista ordinata) sono gestite come le liste semplici viste nella dispensa 17, solo che gli elementi all'interno della lista vengono mantenuti ordinati rispetto un criterio gestito direttamente in fase di inserimento di un nuovo elemento.

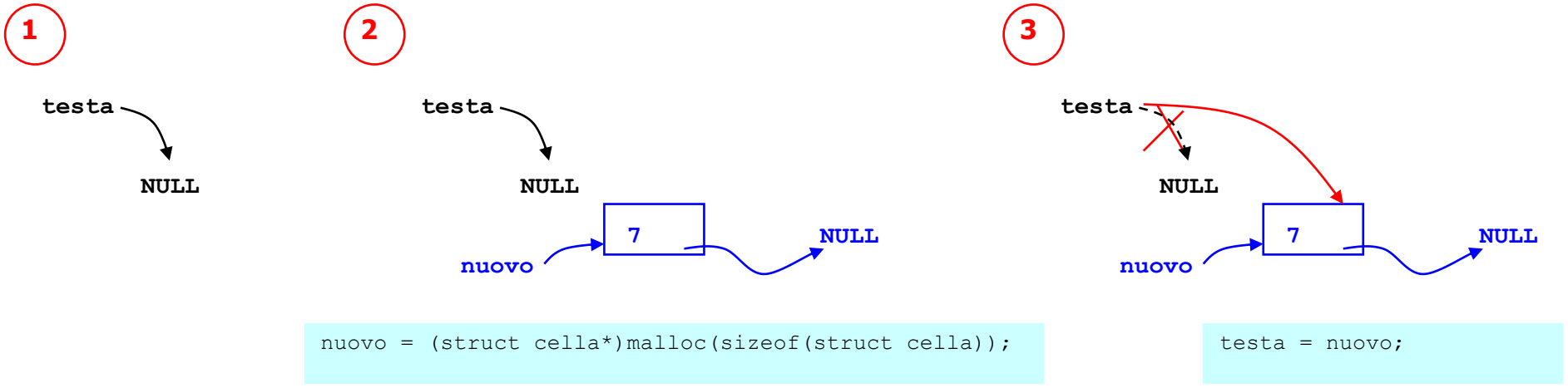
Immaginiamo per esempio l'ufficio accettazione di un pronto soccorso, i vari malati vengono messi in fila per la visita in ordine di arrivo e soprattutto in ordine di gravità, quindi un malato grave sicuramente verrà visitato prima di uno meno grave anche se arrivato prima al pronto soccorso.

La struttura dati necessaria per gestire la lista con priorità è la stessa vista per le liste semplici:

```
struct cella{
      int ordine;
      struct cella *next;  //puntatore all'elemento successivo
};

struct cella *testa = NULL;
```

Consideriamo nel nostro esempio un valore interno all'interno di ogni cella per gestire la priorità (o l'ordine). L'obiettivo è quello di mantenere la lista ordinata in maniera crescente ad ogni inserimento.

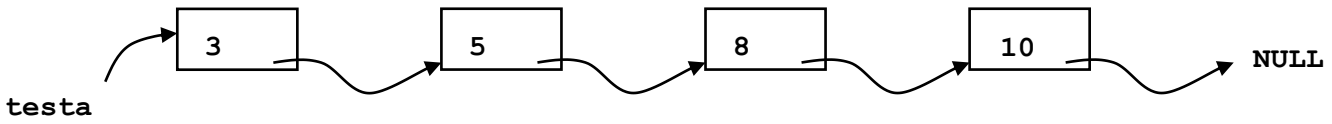
Il caso più semplice si verifica quando la lista è vuota, ovviamente il nuovo elemento da inserire indipendentemente dal valore della priorità verrà inserito in testa:



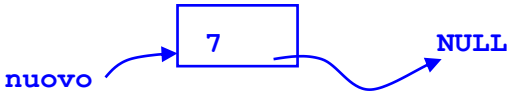

```
nuovo = (struct cella*)malloc(sizeof(struct cella));
```

```
testa = nuovo;
```

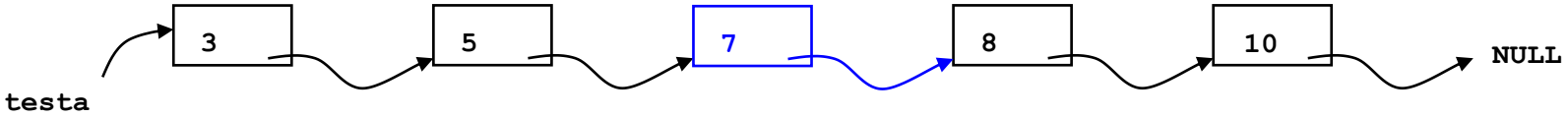
Consideriamo una lista con vari elementi:



Se il nuovo elemento da inserire ha come ordine il numero 7



Ovviamente dovrebbe essere inserito tra la cella con il valore 5 e la cella con il valore 8:

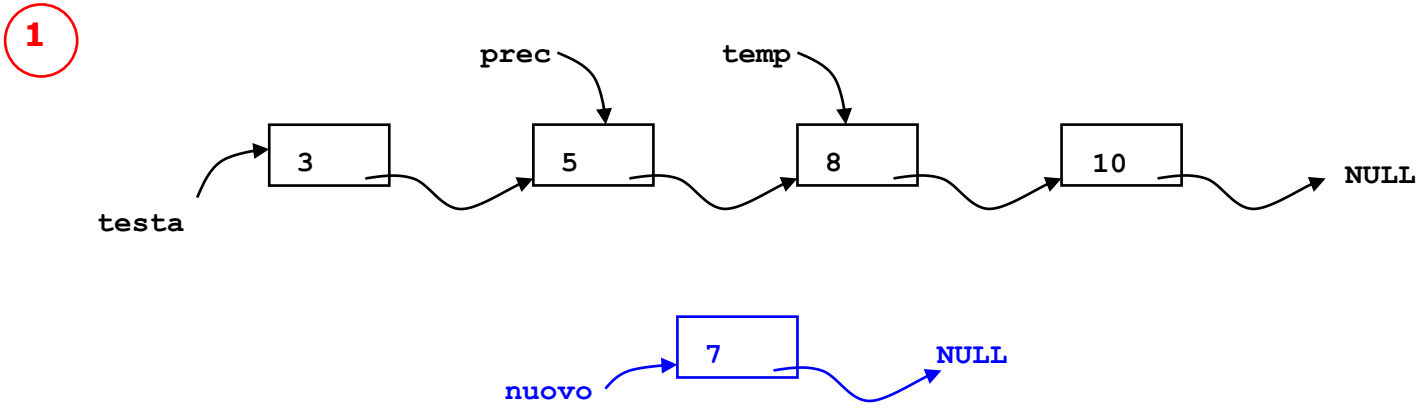


Per inserire la nuova cella nella posizione giusta dovremo scorrere la lista fino ad arrivare alla prima cella con valore maggiore rispetto quella che deve essere inserita. Dato che la lista è ordinata in maniera crescente, il primo elemento maggiore sarà anche quello che dovrà stare immediatamente dopo al nuovo elemento dopo l'inserimento.

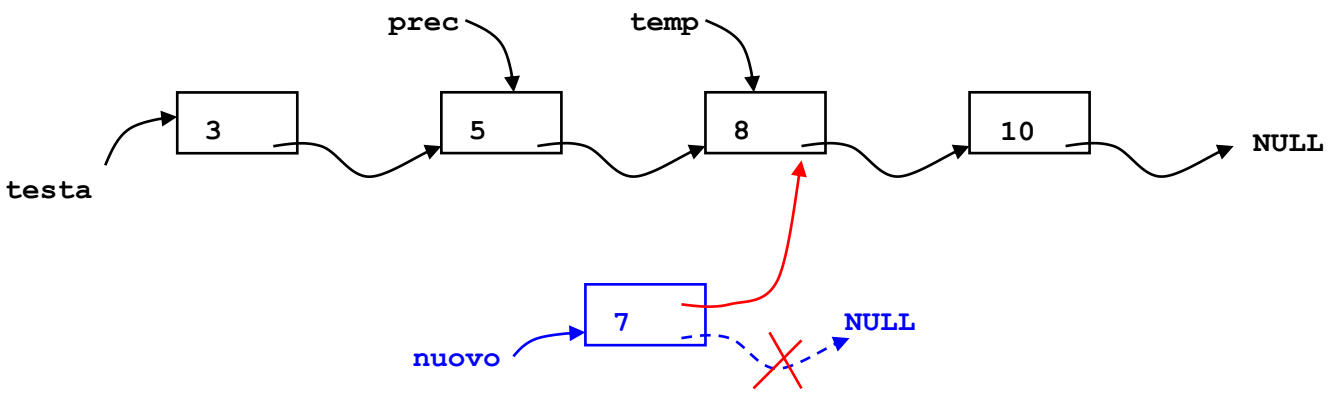
```
nuovo = (struct cella*)malloc(sizeof(struct cella));
nuovo->valore = valoreDaInserire;

if (testa == NULL) //lista vuota
      testa = nuovo;
else
{
      prec = NULL;
      temp = testa;
      while (temp != NULL)
```

```
            {
                  if (temp->valore > nuovo->valore) //trovata la cella con valore più grande
                  {
                        nuovo->next = temp;
                        if (prec == NULL) // se l'elemento è da inserire in testa
                              testa = nuovo;
                        else
                              prec->next = nuovo;
                        break;
                  }
                  prec = temp;
                  temp = temp->next;
            }
      }
```

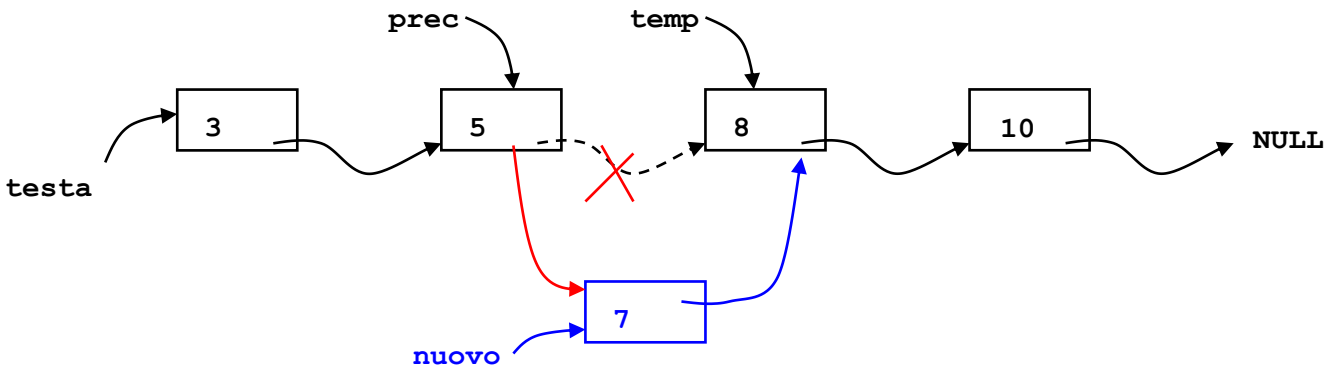
Quando si arriva all'interno del corpo dell'istruzione if all'interno del while (quindi quando viene trovata la cella con valore più alto rispetto a quella da inserire), la situazione sarà la seguente:



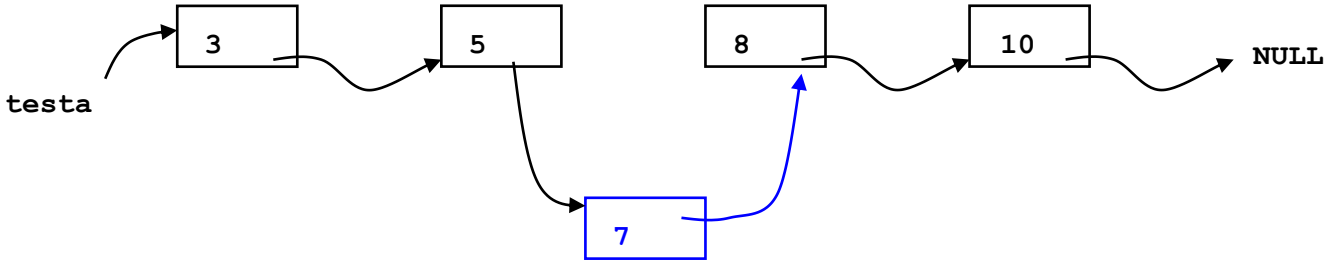
2


prec
temp
3
5
8
10
NULL
testa
7
NULL
nuovo

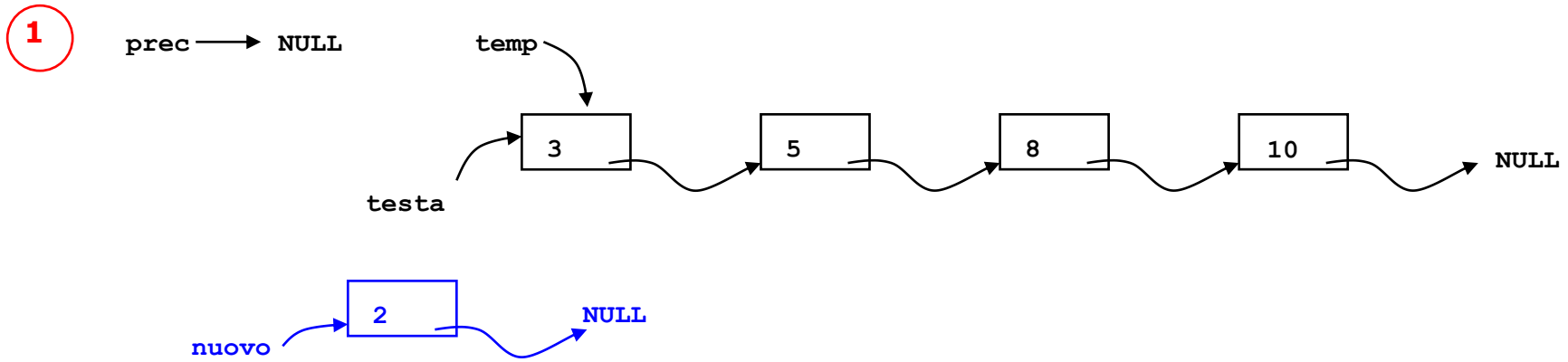
3

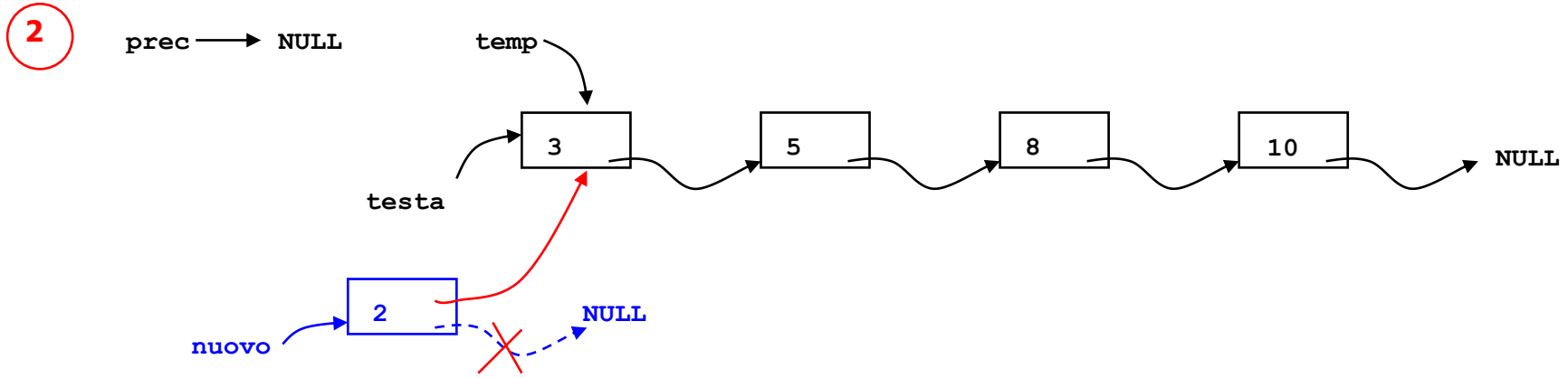

prec
temp
3
5
8
10
NULL
testa
7
nuovo

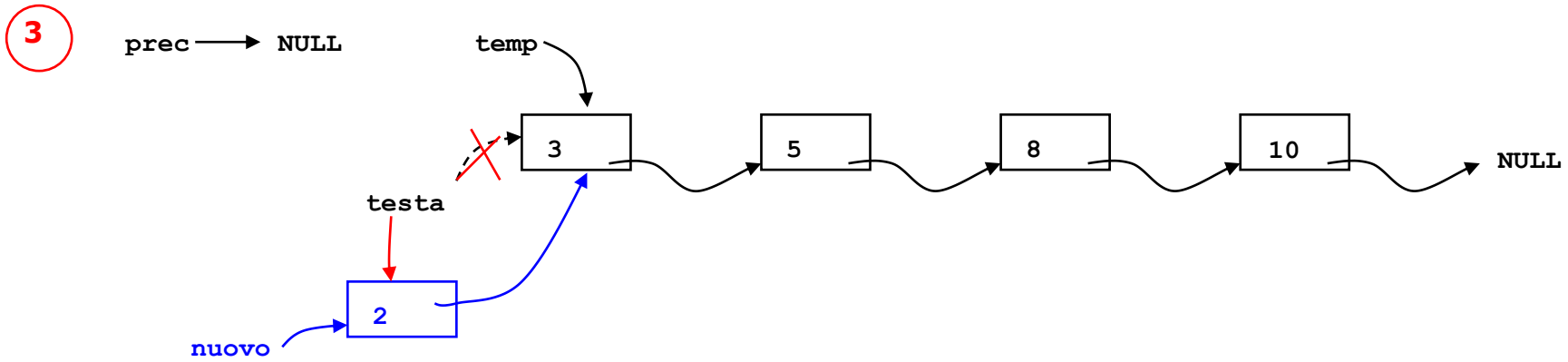
4

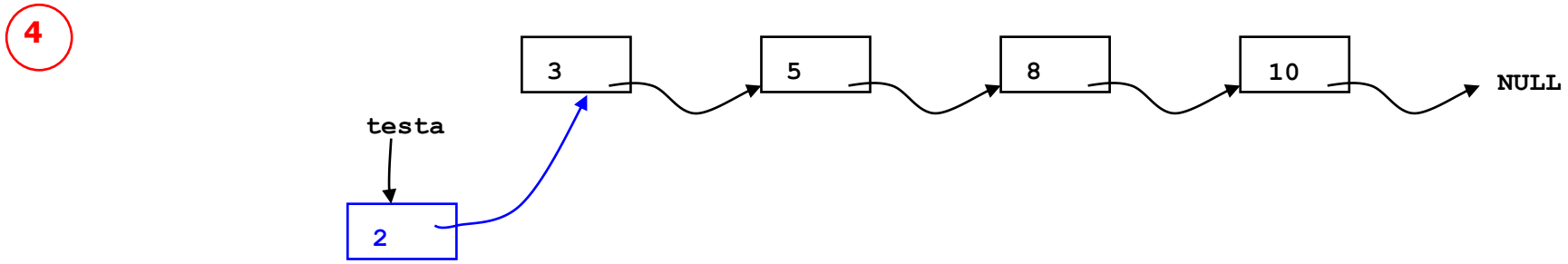

3
5
8
10
NULL
testa
7

Se l'elemento deve essere inserito in testa:

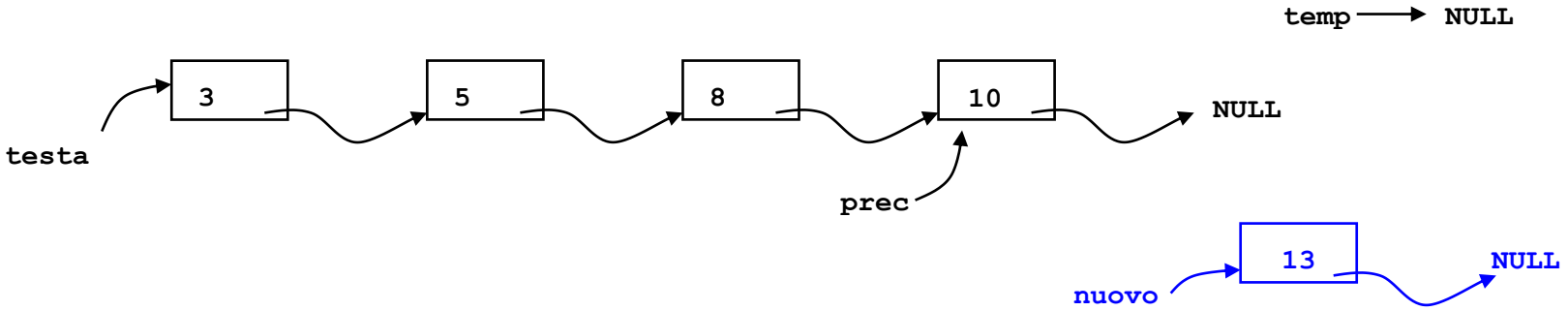






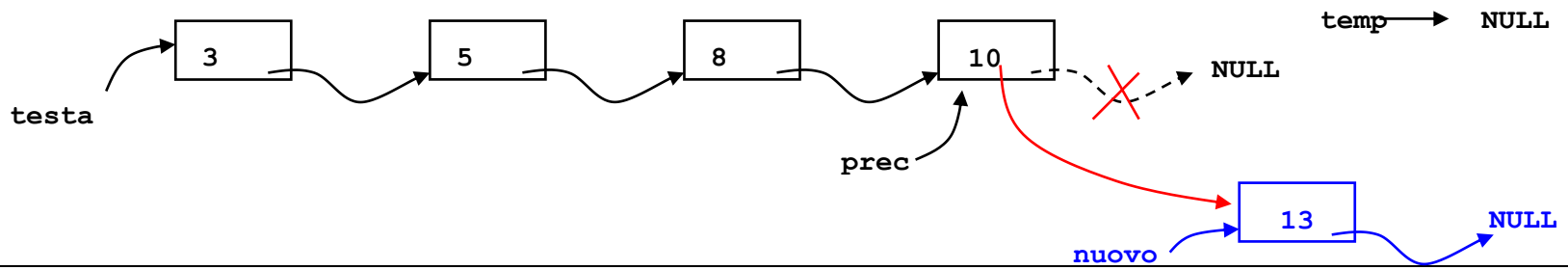


Se l'elemento deve essere inserito in coda, l'algoritmo descritto sopra non funziona in quanto all'interno del ciclo while non si troverà mai una cella più grande rispetto a quella da inserire:

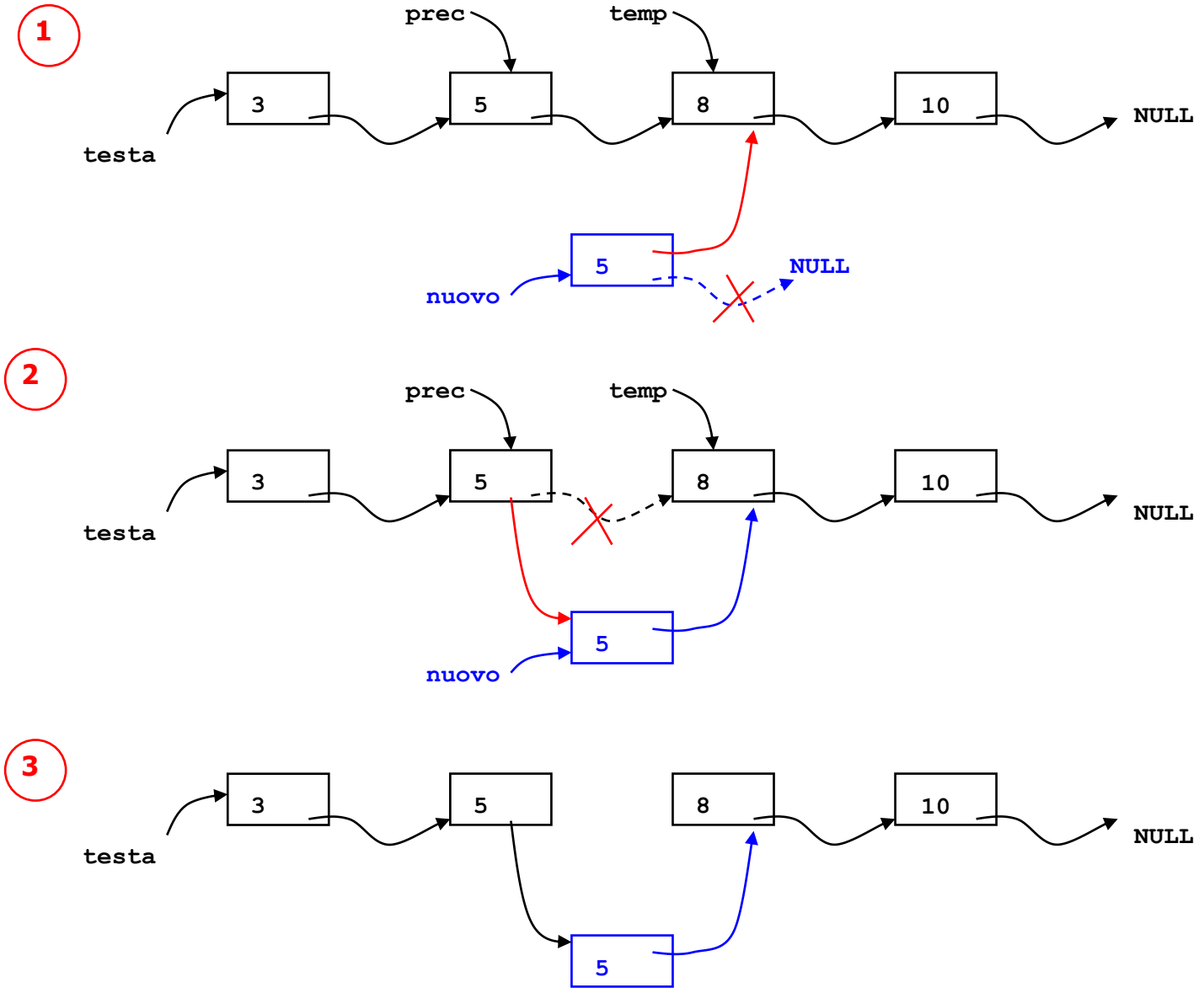


Quindi dopo il ciclo bisognerà fare un controllo per capire se l'elemento non è stato inserito e quindi ci troviamo in fondo alla lista:

```
nuovo = (struct cella*)malloc(sizeof(struct cella));
nuovo->valore = valoreDaInserire;

if (testa == NULL) //lista vuota
      testa = nuovo;
else
{
      prec = NULL;
      temp = testa;
      while (temp != NULL)
      {
            if (temp->valore > nuovo->valore) //trovata la cella con valore più grande
            {
                  nuovo->next = temp;
                  if (prec == NULL) // se l'elemento è da inserire in testa
                        testa = nuovo;
                  else
                        prec->next = nuovo;
                  break;
            }
            prec = temp;
            temp = temp->next;
      }

      if (temp == NULL) //il nuovo elemento non è stato inserito e siamo in fondo alla lista
            prec->next = nuovo;
}
```



Considerando l'ultimo algoritmo, nel caso in cui il valore della nuova cella da inserire sia già presente all'interno della lista, la nuova cella viene inserita dopo quella con il valore uguale in quanto all'interno del ciclo si cerca la cella con un valore maggiore rispetto a quella da inserire:



Se invece la nuova cella deve essere inserita prima di quella con il valore uguale, l'unica modifica da fare nell'algoritmo è nell'espressione presente nell'istruzione if nel corpo del ciclo while: sostituire il simbolo maggiore (>) con maggiore uguale (>=)

Nel caso in cui la lista deve essere ordinata in modo decrescente basta inserire, sempre nel controllo dell'istruzione if il minore (<) o minore uguale, il resto rimane invariato!

Tutte le altre operazioni che possono essere eseguite sulla lista ordinata sono fatte allo stesso modo delle liste semplici come descritto nella dispensa 17.