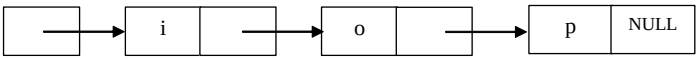
ESERCIZIO 1

Sia data la struttura seguente:

```
struct elem {char info; elem* pun;};
```

Scrivere una funzione che prende in ingresso una stringa `st` e restituisce una lista di elementi di tipo `elem`. La lista restituita deve contenere un elemento per ogni carattere diverso della stringa `st` (la lista non deve contenere duplicati) e deve essere ordinata in ordine crescente per il valore del campo informazione degli elementi.

Per esempio, se la funzione viene chiamata con la stringa "`pippo`", la lista restituita è la seguente:



ESERCIZIO 2

Scrivere una funzione che prende in ingresso il nome di un file, legge interi dal file terminati dal carattere '.' e restituisce una matrice quadrata di dimensione 8x8 costruita come segue: diagonale principale inizializzata con il massimo dei valore letti; diagonale secondaria inizializzata con il minimo dei valori letti; tutti gli altri elementi inizializzati con la somma dei numeri letti.

Se viene rilevato un errore durante la lettura dal file oppure il file ha come primo elemento il carattere '.', la matrice restituita deve essere inizializzata con tutti gli elementi uguali a 0.

Per esempio, se la funzione viene chiamata con il file di contenuto:

3 5 1 .

la matrice restutuita è la seguente:

```
5 9 9 9 9 9 9 1
9 5 9 9 9 9 1 9
9 9 5 9 9 1 9 9
9 9 9 5 1 9 9 9
9 9 9 1 5 9 9 9
9 9 1 9 9 5 9 9
9 1 9 9 9 9 5 9
1 9 9 9 9 9 9 5
```

ESERCIZIO 3

Scrivere una funzione ricorsiva che dato un vettore di interi, passato come argomento alla funzione, restituisce `true` se il vettore è ordinato in ordine decrescente, `false` altrimenti. La funzione deve poter essere chiamata con vettori di varie dimensioni.

SOLUZIONI

Esercizio 1

```
void insordinata(elem*& p0, char c) {
          elem* p; elem* q; elem* r;
          for (q = p0; q != NULL && q->info < c; q = q->pun)
                    p = q;
          if (q!= NULL && q->info == c)
            return;

          r = new elem;
          r->info = c;
          r->pun = q;
          // controlla se si deve inserire in testa
          if (q == p0)
          p0 = r;
          else p->pun = r;
  }

elem* crealista(const char * st){
      elem* testa=NULL;
      for (int i=0; st[i] != '\0'; i++)
       insordinata(testa, st[i]);
      return testa;
}
```

Esercizio 2

```
int* crea_matrice(const char* nome) {
   cont int n = 8;
   int num, min, max, somma;
   int* mat = new int [n*n];
   for (int i=0; i< n*n; i++)
            mat[i]=0;
   ifstream leggi(nome);
   leggi >> num;
   if(!leggi)
      return mat;
   min=max=somma=num;
   while(leggi>>num) {
     max = (num > max)?num:max;
     min = (num < min)?num:min;
     somma+=num;
   }
```

```
 leggi.clear();
 char c;
 if (leggi>>c && c == '.'){
    for(int i=0; i< n; i++)
      for (int j=0; j<n; j++)
        if (i==j)
           mat[i*n+j] = max;
        else if(j == n-1-i)
              mat[i*n+j] = min;
            else mat[i*n+j] = somma;
 }
leggi.close();
return mat;
}
```

Esercizio 3

```
bool controlla(int* v, int n) {
 if (n==1)
    return true;
 if (v[n-1] > v[n-2])
      return false;
 return  controlla(v, n-1);
}
```