Esercizio 3AIIN 30.03.2021 Ordinamento doppio

Scrivere una procedura che ordina gli studenti di una classe in base al peso e, a parità di peso, in base all'altezza.

Nota: peso ed altezza sono due esempi di chiavi secondarie.

```
#define MAX_STUDENTI 30

typedef struct{
	char cognome[25];
	char nome[21];
	double peso;
	double altezza;
}studente;


int main(){
studente classe[MAX_STUDENTI];
int numStudenti = 0;


}
```

```
void sortDoppio(studente classe[], int numStudenti){  //I/O, I
int i, j;
        for(i=0; i < numStudenti -1 ; i++)
           for(j=i+1; j < numStudenti; j++)
                if( (classe[i].peso > classe[j].peso)                          //i pesi non sono in ordine
                        || ((classe[i].peso == classe[j].peso)                 //i pesi sono uguali
                                && (classe[i].altezza > classe[j].altezza) ) ){      //e le altezze non sono ordinate

                        studente temp;
                        temp = classe[i];
                        classe[i] = classe[j];
                        classe[j] = temp;
                }
}
```

Esercizio Fusione (merge) di 2 vettori (ordinati)

Dati due vettori di record ordinati per codice prodotto (codProd) in modo crescente, scrivi un sottopgm **merge**() che li fonde in un terzo vettore.

Si veda l'esempio.

```
typedef struct{
        char codProd[6];                //chiave primaria
        unsigned int quantita;
}prodotto;
```

magaz1		**magaz2**		**magazFus**	
codProd	quantità	codProd	quantità	codProd	quantità
1	2	2	1	1	2
3	1	**3**	2	2	1
5	2	7	2	**3**	3
6	1	**8**	3	5	2
8	2	14	1	6	1
12	3	**15**	3	7	2
15	1	22	0	**8**	5
		25	3	12	3
		37	2	14	1
		39	2	**15**	4
		50	1	22	0
				25	3
				37	2
				39	2
				50	1

```
void merge(const prodotto magaz1[], int numProdMagaz1,
           const prodotto magaz2[], int numProdMagaz2,
           prodotto magazFus[], int & numProdMagazFus);
```

Esercizio Disegno dello stack

Scrivi un pgm C++ che utilizza una funzione **contiene()** (deve essere codificata) che prende in input due stringhe e restituisce 1 se tutti i caratteri della prima stringa sono contenuti nella seconda stringa, 0 in caso contrario.
La funzione utilizza al suo interno una funzione **ricerca()** (deve essere codificata).

Esempi:

contiene("prova","ora")	restituisce 0
contiene("alba","ballare")	restituisce 1
contiene("aaa","ora")	restituisce 1
contiene("ora","")	restituisce 0

```
#define DIM_STR 30 +1

int main(){
char str1[DIM_STR];
char str2[DIM_STR];

	cout<<"Digita la prima stringas\n";
	cin.getline(str1,DIM_STR);

	cout<<"Digita la seconda stringa\n";
	cin.getline(str2,DIM_STR);

	if(contiene(str1,str2))
		cout<<"Tutti i caratteri dalla prima stringa sono contenuti nella seconda stringa\n";
	else
		cout<<"NON tutti i caratteri dalla prima stringa sono contenuti nella seconda stringa\n";

	return 0;
}
```

Disegna lo stack.

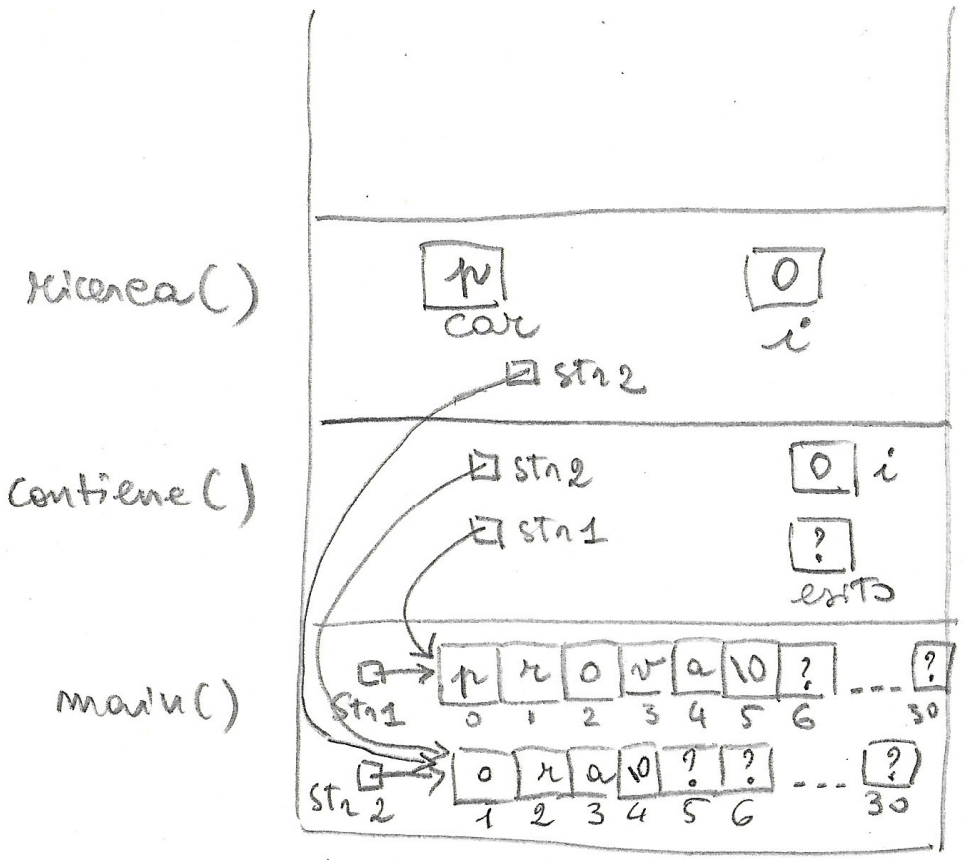
```
int contiene(const char str1[], const char str2[]){     //I, I
int i;
int esito;

	 for(i=0; str1[i]; i++){
		esito=ricerca(str1[i],str2);
		if(esito==0)
			return 0;
	 }

	return 1;
}
```

```
int ricerca(char car, const char str2[]){          //I,I
int i;

        for(i=0; str2[i]; i++)
                if(car==str2[i])
                        return 1;
        return 0;
}
```



fotografia dello stack all'inizio della 1a chiamata della funzione ricerca().