ESERCIZIO STAZIONI DI SERVIZIO

Giovanni deve percorrere M chilometri in motocicletta. Prima di partire si segna la posizione delle n stazioni di servizio s_0, s_1, . . . , s_{n-1} presenti lungo il percorso. Tali posizioni sono identificate dalle distanze (in chilometri) d_0, d_1, . . . d_{n-1} dove d_0 è la distanza dal punto di partenza alla stazione s0, e per i = 1, . . . , n − 1 è la distanza fra le stazioni s_{i-1} e s_i. Inoltre, per i = 0, . . . , n − 1, pi indica il prezzo (al litro) praticato nella stazione s_i. La motocicletta consuma 0.05 litri per chilometro e ha un serbatoio di 30 litri inizialmente pieno. Giovanni decide di riempire totalmente il serbatoio ogni volta che si ferma in una stazione di servizio. Scrivere una procedura

```
void pianifica(double M, int n, double *d, double *p, int s, double dist, piano *c, piano
*b);
```

che dati gli array delle distanze `d` e dei prezzi `p` stampi l'elenco delle stazioni di servizio in cui si deve fermare per minimizzare la spesa per il carburante (si ignori il valore del carburante che rimane nel serbatoio al termine del viaggio).

I parametri passati alla funzione sono dettagliati di seguito:

`M`: km totali da percorrere,
`n`: numero delle stazioni lungo il percorso,
`d`: array delle distanze tra le stazioni,
`p`: prezzo del carburante nelle stazioni,
`s`: posizione/stazione corrente (a che livello dell'albero di backtracking si trova la funzione corrente),
`dist`: distanza percorsa attualmente
`c`: piano attuale
`b`: miglior piano

Inoltre il tipo `piano` è definito come segue:

```
typedef struct {
            double costo;
            int *stazione;
} piano;
```

dove `costo` è il costo del piano e `stazione` è un puntatore ad un array binario (0-1) che contiene la lista delle fermate effettuate (1) e non (0).

```
int array di fermate
```