

*/\* Completare il programma C nelle parti indicate con TODO (lasciando inalterate le altre parti) in modo che:*

*legga da standard input un numero n, SIZE valori per la white list e SIZE valori per la black list (SIZE è uguale a 5);*

*stampi la lista dei divisori, separati da uno spazio, di n presenti nella white list e non presenti nella black list*

*Suggerimento:*

*per provare il programma sul proprio PC senza dover digitare ogni volta gli 11 valori richiesti, creare un file di testo (values.txt) con gli 11 valori e lanciare il comando da terminale:*

*./a.out < values.txt*

*Es. di file values.txt (il primo valore è n, seguono 5 valori per la white liste e 5 valori per la black list):*

```
120
2
3
4
7
10
2
10
20
30
40
```

*In questo caso l'output sarà:*

*I divisori primi di 120 presenti nella white list e non nella black list sono:*

```
3
```

*\*/*

```
#include <stdbool.h>
```

```
#include <stdio.h>
```

```
#define SIZE 5
```

```
void print_dividers(int n, int white_list[], int size_white, int black_list[], int size_black);
```

```
bool in(int n, int values[], int size);
```

```
bool prime_r(int n, int i);
```

```
bool prime_i(int n);
```

*/\**

*\* Program entry point*

*\*/*

```
int main() {
```

```
    int n; //120
```

```
    int white_list[SIZE];
```

```
    int black_list[SIZE];
```

```
    int i = 0; // w_L
```

```
    int j = 0; // B_L
```

```
    scanf("%d", &n);
```

```

// Popola white_list, acquisendo dallo standard input SIZE valori per
// TODO

for( i = 0; i < SIZE ; i++ ){
    scanf("%i", &white_list[i]);
}

// Popola black_list, acquisendo dallo standard input SIZE valori per
// TODO

for( j = 0; j < SIZE; j++ ){
    scanf("%i", &black_list[j]);
}

printf("I divisori primi di %d presenti nella white list e non nella black list sono:
\n", n);
print_dividers(n, white_list, SIZE, black_list, SIZE);
printf("\n");
return 0;
}

/*
 * Stampa i divisori primi di n, seguiti da uno spazio,
 * che sono presenti in white_list e non sono presenti in black_list.
 * Usa la funzione in() per verificare la presenza in una lista
 * e la funzione ricorsiva prime_r() per verificare se un numero è primo.
 * L'eventuale uso della funzione iterativa prime_i() avrà una valutazione inferiore.
 */
void print_dividers(int n, int white_list[], int size_white, int black_list[], int
size_black) {
    // TODO

    int i;
    int resto;
    bool divisore;
    bool in_black_list;
    bool is_prime;

    for(i = 0; i < size_white ;i++ ){

        resto = n % white_list[i];
        //divisore = resto == 0;

        if( resto == 0 ){

            in_black_list = in( white_list[i], black_list, size_black);    //salviamo il
valore di return in in_black_list

            if( in_black_list == false ){

                is_prime = prime_i(white_list[i]);

                if( is_prime == true){

                    printf("%i ", white_list[i]);
                }
            }
        }
    }
}

```

```

    }
}

/*
 * Restituisce true se n è presente in values, false altrimenti,
 * utilizzando la ricorsione.
 * L'eventuale soluzione iterativa avrà una valutazione inferiore.
 */
bool in(int n, int values[], int size) {           // controllo se n è presente in B_L
    // TODO

    int i = 0;

    for( i = 0; i < size; i++ ){                   //scorro l'array B_L / values
        if( n == values[i] ){
            return true;                           //return true termina la funzione
        }
    }

    return false;                                 //return false fa terminare la funzione
}

bool in_r(int n, int values[], int size) {         // controllo se n è presente in B_L
    // TODO

    if( size < 0 ){
        return false;
    }

    if( n == values[size - 1] ){
        return true;
    }

    return in_r(n, values, size - 1);
}

```

```

/*
 * Dato n >= 0 restituisce true se è primo, false altrimenti,
 * utilizzando la ricorsione.
 */
bool prime_r(int n, int i) {                       //se W_L[i] è primo
    // TODO

    if( i == 1 ){
        return true;
    }

    if( n % i == 0 ){
        return false;
    }

```

```
    } else{
        return prime_r(n , i-1);        //passo ricorsivo
    }
}

/*
 * Dato n >= 0 restituisce 1 se è primo, 0 altrimenti,
 * utilizzando l'iterazione.
 * ATTENZIONE: la funzione va implementata soltanto se non si è implementata prime_r()
 */
bool prime_i(int n) {
    // TODO

    int i;

    for( i = 2; i < n ; i++ ){
        if ( n % i == 0){
            return false;        //ho trovato un divisore    -->    non è primo
        }
    }
    return true;                //true se è primo    -->    non ho mai trovato un divisore
}
```