Teoria

1. Dato il seguente array tridimensionale: int A[2][3][4];

Scrivi un'espressione che, usando l'aritmetica dei puntatori, acceda all'elemento A[1][2][3].

2. Considera il seguente codice:

int x = 5; int *p = &x; int **q = &p; int ***r = &q;

Cosa stamperanno le seguenti istruzioni? Spiegare perché.

printf("%d\n", **q);

printf("%p\n", *r);

printf("%d\n", ***r + 2);

3. Analizza il seguente codice:

int* foo() { static int x = 10; return &x; }

int main() { int *p = foo(); printf("%d\n", *p); return 0; }

a) Questo codice crea un dangling pointer? Perché o perché no?

b) Quali sono le differenze tra questo codice e quello che crea un dangling pointer visto negli esercizi precedenti?

c) Quali sono i potenziali problemi o vantaggi di questo approccio?

4. Considera il seguente codice:

```
int p = (int)malloc(sizeof(int) * 5);

int *q = p + 2;

free(p);

*q = 10;
```

a) Dopo l'esecuzione di free(p), cosa rappresenta q?

b) Quali sono i rischi associati all'uso di q dopo la chiamata a free(p)?

c) Come si potrebbe modificare questo codice per evitare potenziali problemi di memoria?

5. Dato il seguente codice:

```
int arr[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}};

int (*p)[3] = arr + 1;

printf("%d", (*p)[1]);
```

Cosa verrà stampato?

a) 4

b) 5

c) 6

d) 7

6. Considera il seguente frammento di codice:

```
int arr[3][4] = {{1,2,3,4}, {5,6,7,8}, {9,10,11,12}};
int (*p)[4] = arr;
```

Cosa rappresenta p? Come si accederebbe all'elemento con valore 7 usando p?

7. Analizza la seguente dichiarazione:

```
int (*func)(int *, int (*)(int, int));
```

Cosa rappresenta func? Descrivi in dettaglio il tipo di funzione che può essere assegnato a func.

Esercizi

1. Scrivi una funzione che ruoti una matrice quadrata di 90 gradi in senso orario:

```
void rotate_matrix(int** matrix, int n);
```

Input:

1 2 3

4 5 6

7 8 9

Output:

7 4 1

8 5 2

9 6 3

2. Implementa una funzione che trova la sottostringa palindroma più lunga in una stringa data:

```
char* longest_palindrome_substring(const char* s);
```

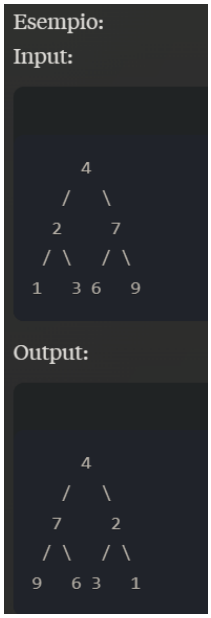
Esempio: Input: "babad" Output: "bab" o "aba"

3. Implementa una funzione che converta un BST ad una lista:

```
struct BST* convertBSTtoList(struct BST* root)
```

4. Implementare una funzione che inverta un albero binario. La funzione deve avere la seguente firma:

```
struct BST* invertTree(struct BST* root);
```



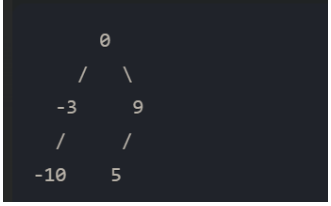
5. Implementare una funzione che costruisca un albero binario di ricerca bilanciato da un array di interi ordinato in ordine crescente. La funzione deve avere la seguente firma:

```
struct BST* sortedArrayToBST(int* nums, int numsSize);
```

Esempio:

Input: nums = [-10,-3,0,5,9]

Output:



6. Implementare una funzione che unisca k liste concatenate ordinate in una singola lista ordinata. La funzione deve avere la seguente firma:

```
struct ListNode* mergeKLists(struct ListNode** lists, int listsSize);
```

Esempio:

Input:

```
[
  1->4->5,
  1->3->4,
  2->6
]
```

Output: 1->1->2->3->4->4->5->6

7. Implementare una funzione che inverta una sottolista di una lista concatenata data, dal nodo in posizione m al nodo in posizione n. La funzione deve avere la seguente firma:

```
struct ListNode* reverseBetween(struct ListNode* head, int m, int n);
```

Esempio:

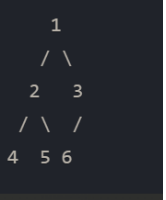
Input: 1->2->3->4->5->NULL, m = 2, n = 4

Output: 1->4->3->2->5->NULL

8. Implementare una funzione che determini se un albero binario dato è completo. Un albero binario completo è un albero in cui ogni livello, eccetto possibilmente l'ultimo, è completamente riempito, e tutti i nodi sono il più a sinistra possibile. La funzione deve avere la seguente firma:

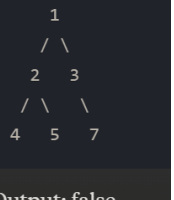
```
bool isCompleteTree(struct BST* root);
```

Esempio 1:
Input:



Output: true

Esempio 2:
Input:



Output: false

9. Implementare una funzione che determini se una parola esiste in una griglia di lettere. La parola può essere costruita da lettere di celle sequenzialmente adiacenti, dove le celle adiacenti sono quelle orizzontalmente o verticalmente vicine. La stessa cella della lettera non può essere usata più di una volta. La funzione deve avere la seguente firma:

```
bool exist(char** board, int boardSize, int* boardColSize, char* word);
```

Esempio:
board =

```
[
  ['A','B','C','E'],
  ['S','F','C','S'],
  ['A','D','E','E']
]
```

word = "ABCCED"
Output: true

word = "SEE"
Output: true

word = "ABCB"
Output: false