Test 5

1. In C++, cosa definisce la dichiarazione `const int* ptr;`?
    1. Un puntatore costante a un intero.
    2. Un puntatore a un intero costante.
    3. Un puntatore costante a un intero costante.
    4. La dichiarazione non è valida.
2. Per quale motivo è importante dichiarare `virtual` il distruttore di una classe base se si prevede di usare il polimorfismo?
    1. Per evitare che la classe base possa essere istanziata.
    2. Per garantire che venga chiamato anche il distruttore della classe derivata quando si usa `delete` su un puntatore alla classe base.
    3. Per rendere più veloce l'allocazione di memoria per gli oggetti derivati.
    4. Un distruttore non può mai essere `virtual`.
3. Quale delle seguenti affermazioni descrive meglio l'allocazione di memoria sullo "Stack"?
    1. È gestita manualmente dal programmatore tramite `new` e `delete`.
    2. È usata per le variabili globali e ha una durata pari a quella del programma.
    3. È gestita automaticamente, usata per variabili locali e parametri di funzione, e la memoria viene liberata quando escono dallo scope.
    4. È più lenta dell'allocazione sull'Heap.
4. Qual è la complessità temporale media di inserimento e ricerca in un `std::map` o `std::set` in C++?
    1. O(1)
    2. O(logn)
    3. O(n)
    4. O(nlogn)
5. Nella cancellazione di un nodo con due figli da un Albero Binario di Ricerca (BST), quale nodo viene tipicamente usato per sostituire il nodo da eliminare?
    1. Il nodo più a sinistra nel sottoalbero sinistro.
    2. Qualsiasi nodo foglia.
    3. Il predecessore in-ordine (il più grande nel sottoalbero sinistro) o il successore in-ordine (il più piccolo nel sottoalbero destro).
    4. La radice dell'albero.
6. Cosa fa il seguente ciclo?

    C++

    ```
    int n = 100;
    int count = 0;
    for (int i = 1; i < n; i = i * 2) {
        count++;
    }
    ```

    1. Il ciclo non termina mai.
    2. `count` conterrà il valore di `n`.
    3. Esegue un numero di iterazioni logaritmico rispetto a `n`.
    4. Esegue un numero di iterazioni lineare rispetto a `n`.
7. Qual è la differenza principale tra `size()` e `capacity()` di un `std::string` o `std::vector`?
    1. Non c'è alcuna differenza, sono sinonimi.
    2. `size()` è il numero di elementi contenuti, `capacity()` è la quantità di memoria allocata prima che sia necessaria una riallocazione.

    3. `capacity()` è sempre uguale a `size()`.
    4. `size()` è la memoria allocata in byte, `capacity()` è il numero di elementi.
8. L'ereditarietà multipla in C++ si riferisce a:
    1. Una classe che eredita da un'altra classe, la quale a sua volta eredita da una classe base.
    2. Una classe che serve da base per molte altre classi.
    3. Una classe che eredita da più di una classe base direttamente.
    4. Più oggetti che ereditano le proprietà di un singolo oggetto.
9. Quale operatore di cast in C++ è utilizzato per conversioni potenzialmente non sicure, come castare un tipo di puntatore a un altro tipo di puntatore non correlato?
    1. `static_cast`
    2. `dynamic_cast`
    3. `const_cast`
    4. `reinterpret_cast`
10. Se una funzione riceve un oggetto passato "per valore", quale funzione membro di quell'oggetto viene invocata?
    1. Il costruttore di default.
    2. Il distruttore.
    3. Il costruttore di copia.
    4. L'operatore di assegnamento.
11. Quale di queste strutture dati implementa una politica LIFO (Last-In, First-Out)?
    1. BST
    2. Coda
    3. Stack
    4. Lista
12. Cosa si intende per "overloading" (sovraccarico) di una funzione?
    1. Ridefinire una funzione di una classe base in una classe derivata.
    2. Definire più funzioni con lo stesso nome ma con parametri diversi (per tipo o per numero).
    3. Definire una funzione che può accettare un numero variabile di argomenti.
    4. Creare un alias per una funzione esistente.
13. Come si apre un file binario in modalità scrittura in C++?
    1. `ofstream file("data.bin", ios::in);`
    2. `ofstream file("data.bin", ios::binary);`
    3. `fstream file("data.bin", ios::app);`
    4. `ifstream file("data.bin", ios::binary);`
14. Quale delle seguenti affermazioni riguardo ai riferimenti (`&`) è FALSA?
    1. Un riferimento deve essere inizializzato al momento della dichiarazione.
    2. Un riferimento non può essere `nullptr`.
    3. Un riferimento può essere fatto puntare a un altro oggetto dopo l'inizializzazione.
    4. Un riferimento agisce come un alias per un oggetto esistente.
15. Qual è la complessità nel caso peggiore della cancellazione dell'ultimo elemento in una lista concatenata semplice se si ha solo il puntatore alla testa?
    1. O(1)
    2. O(logn)
    3. O(n)
    4. O(n2)
16. Nel contesto delle classi, a cosa serve il puntatore `this`?
    1. A puntare a una variabile globale.
    2. A puntare all'istanza dell'oggetto corrente all'interno di un suo metodo non statico.
    3. A essere usato solo nelle funzioni `static`.

    4. A puntare all'indirizzo della funzione `main`.
17. Una classe che contiene almeno una funzione virtuale pura è detta:
    1. Classe template
    2. Classe derivata
    3. Classe astratta
    4. Classe amica (friend class)
18. Qual è l'output del seguente codice?

    C++

    ```
    #include <iostream>
    using namespace std;

    void func(int n) {
        if (n == 0) return;
        cout << n % 2;
        func(n / 2);
    }

    int main() {
        func(13);
    }
    ```

    1. 1101
    2. 1011
    3. 13
    4. 6
19. Se si ha una Pila (Stack), quale delle seguenti operazioni NON è tipicamente supportata in O(1)?
    1. `push` (inserimento in cima).
    2. `pop` (rimozione dalla cima).
    3. `top` (lettura della cima).
    4. `search` (ricerca di un elemento arbitrario).
20. Qual è la complessità della seguente funzione?

    C++

    ```
    int function(int n) {
        if (n <= 1) return 1;
        return function(n - 1) + function(n - 1);
    }
    ```

    1. O(n)
    2. O(n2)
    3. O(logn)
    4. O(2n)

Test 6

1. A cosa serve la parola chiave `friend` in C++?
    1. A permettere a una classe di ereditare da un'altra.
    2. A creare un alias per un tipo di dato.

    3. A consentire a una funzione o a un'altra classe di accedere ai membri `private` e `protected` della classe che la dichiara amica.
    4. A indicare che un metodo non può modificare lo stato dell'oggetto.
2. Qual è la differenza principale tra il costruttore di copia e l'operatore di assegnamento (`operator=`)?
    1. Non c'è differenza, fanno la stessa cosa.
    2. Il costruttore di copia crea un nuovo oggetto a partire da uno esistente, l'assegnamento sovrascrive un oggetto già esistente.
    3. L'operatore di assegnamento può essere `virtual`, il costruttore di copia no.
    4. Il costruttore di copia è chiamato quando si usa `new`, l'assegnamento quando si usa `delete`.
3. Cosa identifica il seguente frammento di codice?

    C++

    ```
    int* ptr = new int(10);
    delete ptr;
    // ... molto codice dopo ...
    *ptr = 20; // ERRORE
    ```

    1. Memory Leak.
    2. Errore di divisione per zero.
    3. Errore di sintassi.
    4. Uso di un "dangling pointer" (puntatore penzolante).
4. Quale dei seguenti algoritmi di ordinamento ha una complessità nel caso peggiore di O(n2)?
    1. Merge Sort
    2. Quick Sort (con una cattiva scelta del pivot)
    3. Heap Sort
    4. Tutti i precedenti.
5. Dato un puntatore `p` alla testa di una lista, cosa fa la seguente funzione?

    C++

    ```
    void boo(Nodo* p) {
        if (p == nullptr) return;
        boo(p->succ);
        cout << p->valore << " ";
    }
    ```

    1. Stampa la lista dalla testa alla coda.
    2. Stampa la lista in ordine inverso.
    3. Dealloca tutti i nodi della lista.
    4. Causa un ciclo infinito.
6. Quale delle seguenti opzioni è il modo standard per leggere un intero file di testo riga per riga in C++?
    1. `while (!file.eof()) { file >> riga; }`
    2. `for (int i=0; i<100; i++) { getline(file, riga); }`
    3. `while (getline(file, riga)) { ... }`
    4. `file.read(buffer, size);`
7. Se si implementa una Coda (Queue) utilizzando due Pile (Stacks), l'operazione di `enqueue` (inserimento):
    1. È sempre impossibile da realizzare.

2. Può essere realizzata in modo che abbia un costo ammortizzato di O(1).
3. Ha sempre un costo di O(n).
4. È più veloce che implementare una Coda con una lista.

8. Per cosa viene utilizzato `dynamic_cast` in C++?
    1. Per rimuovere la `const`-ezza da una variabile.
    2. Per eseguire conversioni sicure (controllate a runtime) verso il basso in una gerarchia di classi (downcasting).
    3. Per tutte le conversioni tra tipi numerici.
    4. Per rinominare un tipo di dato.
9. Qual è l'output della visita in **pre-ordine** del seguente albero binario?

```
      F
     / \
    B   G
   / \   \
  A   D   I
     / \
    C   E
```

    1. A B C D E F G I
    2. F B A D C E G I
    3. A C E D B I G F
    4. F G I B D E C A
17. In un'implementazione di una Pila (Stack) con un array statico, quale svantaggio principale si presenta?
    1. Le operazioni `push` e `pop` diventano più lente.
    2. La dimensione massima della pila è fissa e limitata.
    3. Si trasforma in una coda.
    4. Non è possibile controllare se la pila è piena.
18. Quale frammento di codice manca alla fine della funzione `inverti()` per completare l'inversione di una lista concatenata?

    C++

```
void inverti(Nodo** testa) {
    Nodo* prev = nullptr;
    Nodo* current = *testa;
    Nodo* next = nullptr;
    while (current != nullptr) {
        next = current->succ;
        current->succ = prev;
        prev = current;
        current = next;
    }
    // AGGIUNGERE RIGA QUI
}
```

    1. `*testa = nullptr;`
    2. `*testa = current;`
    3. `*testa = prev;`
    4. `*testa = next;`
19. Qual è il numero minimo di campi necessari per un nodo di una lista doppiamente concatenata?
    1. 1
    2. 2
    3. 3

4. 4

20. Quale di queste operazioni, su una lista concatenata semplice di lunghezza `n`, ha un costo che dipende da `n` anche se si hanno i puntatori sia al primo che all'ultimo elemento?
    1. Inserire un nuovo elemento in testa.
    2. Aggiungere un nuovo elemento in fondo.
    3. Eliminare il primo elemento.
    4. Eliminare l'ultimo elemento.
21. Qual è la complessità della seguente funzione?

    C++

    ```
    void func(int n) {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j = j + i) {
                cout << "Hi";
            }
        }
    }
    ```

    1. O(n2)
    2. O(nlogn)
    3. O(n)
    4. O(2n)
22. Quando si passa un parametro a una funzione "per riferimento" (`&`), cosa viene effettivamente passato?
    1. Una copia dell'oggetto.
    2. Un puntatore all'oggetto.
    3. Un alias (un altro nome) per l'oggetto originale.
    4. L'indirizzo di memoria dell'oggetto.
23. Quale delle seguenti è una caratteristica fondamentale del polimorfismo in C++?
    1. L'uso di `template` per creare codice generico.
    2. La capacità di definire più costruttori per una classe.
    3. L'associazione a runtime di una chiamata di metodo all'implementazione corretta tramite funzioni `virtual`.
    4. L'incapsulamento dei dati all'interno di una classe.
24. In un Albero Binario di Ricerca (BST), dove si trova l'elemento con il valore più grande?
    1. Sempre alla radice.
    2. Sempre in una foglia a sinistra.
    3. Nel nodo più a destra raggiungibile dalla radice.
    4. Nel nodo più a sinistra raggiungibile dalla radice.
25. Qual è la complessità, nel caso peggiore, per la ricerca di un elemento in una lista concatenata semplice di `n` elementi?
    1. O(logn)
    2. O(n/2)
    3. O(n)
    4. O(1)
26. `cout` in C++ è un'istanza di quale classe?
    1. `iostream`
    2. `ostream`
    3. `istream`
    4. `fstream`
27. Cosa fa la seguente funzione `func()`?

C++

```
int func(Nodo* p) {
    if (p == nullptr) return 1;
    return (p->valore > 0 && func(p->succ));
}
```

1. Restituisce 1 se e solo se la lista è vuota.
2. Restituisce 1 se e solo se tutti gli elementi della lista sono positivi.
3. Restituisce 1 se e solo se la lista è ordinata in modo crescente.
4. Contiene un errore e non può essere compilata.

RISPOSTE

Test 5

1. 2
2. 2
3. 3
4. 2
5. 3
6. 3
7. 2
8. 3
9. 4
10. 3
11. 3
12. 2
13. 2
14. 3
15. 3
16. 2
17. 3
18. 1
19. 4
20. 4

Test 6

1. 3
2. 2
3. 4
4. 2
5. 2
6. 3
7. 2
8. 2

9. 2
10. 2
11. 3
12. 3
13. 4
14. 2
15. 3
16. 3
17. 3
18. 3
19. 2
20. 2