



1) Quale è la relazione fra  $T_1$  e  $T_2$ ?

$$T_1 = O(n \log^3 n)$$

$$T_2 = O((3^n) * n) + (4^n) * n \log n$$

- A)  $T_2 > T_1$   Si vede a occhio che  $T_2 > T_1$
- B)  $T_1 < T_2$   In  $T_2$  vi sono gli esponenti.
- C)  $T_1 = T_2$
- D) Non è possibile determinarlo

2) Quale delle seguenti relazioni contiene almeno un errore?

- A)  $O(1) < O(\log n) < O(\log^2 n)$
- B)  $O(\log n) < O(n \log n) < O(n \log^2 n)$
- C)  $O(1) < O(\log n) < O(n^2)$    $O(n) > O(\log n)$
- D)  $O(\log n) < O(\log^2 n) < O(n^2)$

3) Come è possibile migliorare l'efficienza dell'insertion sort?

- A) Seguendo approssimativamente il pivot
- B) Sfruttando la ricerca Binaria  Migliora l'efficienza
- C) Mediante una sezione casuale dell'elemento iniziale
- D) Nessuna delle precedenti

4) Quale è l'ordine di grandezza (notazione O-grande) della funzione  $f_1$ ?

$$T_1 = 5n^2 + 2n - 10$$

$$T_2 = 5\sqrt{n} + 2$$

$$G = T_1 * T_2$$

A)  $n^{3/2}$

B)  $n\sqrt{n}$

C)  $n^2 + e$

D)  $\sqrt{n}$

$$2\sqrt{5n^2 \cdot n^{1/2}}$$

$$(n^{2+\frac{1}{2}}) = \underline{\underline{n^{5/2}}}$$

5) Quale è la relazione tra  $T_1$  e  $T_2$  (solo n è una variabile)

$$T_1 = O(n \log^2 n)$$

$$T_2 = O(1.5 n \log n)$$

A)  $T_1 > T_2$

B)  $T_1 < T_2$

C)  $T_1 = T_2$

D) Non è possibile determinarlo

6) Quale è l'ordine di grandezza (notazione O-grande) della funzione G?

$$T_1 = 5n^2 + 2n - 10$$

$$T_2 = 5\sqrt{n} + 22$$

$$G = T_1 + T_2$$

$$(n^2) + (\sqrt{n})$$

A)  $5(n^2 + \sqrt{n}) + 2n + 12$

B)  $n^2\sqrt{n}$

C)  $n^2 + e$    $n^2$  pesa di più

D)  $\sqrt{n}$

7) Quale è l'ordine di grandezza (notazione O-grande) della funzione G?

$$T_1 = 5\sqrt{n}^3$$

$$T_2 = 5n\sqrt{n}$$

$$G = T_1 * T_2$$

$$5(\sqrt{n}^3)$$

$$5n(\sqrt{n}^5)$$

)  
(\*)

$$5n(\sqrt{n}^5)$$

- A)  $n^{2.5}$   
B)  $n\sqrt{n}$   
C)  $n^5$    
D)  $n^3\sqrt{n}$

$$25n \left( n^{3/2} \cdot n^{5/2} \right)$$

$$25n(n^4)$$

$$n^{15/2}$$

$$= 25n^5$$

8) Diciamo che  $f(n) = O(g(n))$  se e solo se esistono  $c, n_0 > 0$ :

- A)  $f(n) \leq c g(n) \quad \forall n > n_0 \quad \text{X}$
- B)  $c g(n) \leq f(n) \quad \forall n > n_0$
- C)  $f(n) > c g(n) \quad \forall n > n_0$
- D)  $c f(n) > g(n) \quad \forall n > n_0$

Dalla definizione del  $O(\text{grande})$  si ha che

$$O \leq f(n) \leq c g(n)$$

9) Come è possibile migliorare la retezza dell' Selection Sort

- A) Selezionando il pivot
- B) Sfruttando la Ricerca Binaria Questo è per INSERTION SORT.
- C) //
- D) Nessuna delle precedenti  $\text{X}$  Ha Sempre Complessità  $O(n^2)$

10) Quale è la complessità fra  $T_1$  e  $T_2$  (Solo  $n$  è variabile)

$$T_1 = O((5^e) * n \log n)$$

$$T_2 = O(14^e * n)$$

- A)  $T_1 > T_2$   $\otimes$  questa dato che l'esponenziale ha base maggiore
- B)  $T_1 < T_2$
- C)  $T_1 = T_2$
- D) Non è possibile Determinarla

11) Quale istruzione serve inserire al posto di ??? per stampare il contenuto della matrice?

```
double mat[2][3] = {{1,2,3,4,5,6}};  
for(int f=0; f<2; f++) {  
    cout << ??? << endl;  
    cout << endl;  
}
```

- A)  $mat[f+e]$
- B)  $*(\mathtt{mat} + f)[e]$
- C)  $mat[f]+e$
- D)  $(*(\mathtt{mat} + f) + e) \otimes$

12) Una delle seguenti affermazioni è vera per la funzione Inline

A) Vene eseguita più rapidamente poiché trattata come una riga zero ⊗ Vene sostituto il pezzo di codice.

B)

C)

D)

B) Quale è l'output del seguente codice?

#include <iostream>

using namespace std;

```
Template <typename T>
void fune (const T& x) {
    static int count = 0;
    cout << "x = " << x << "count = " << count << endl;
    # count;
    return;
}
```

```
(# main() {
```

```
    fune<int>(1);
    cout << endl;
```

```
    fune<int>(2);
    cout << endl;
```

```
    fune<double>(1.1);
    cout << endl;
```

```
    return 0;
}
```

A)  $x = 1 \quad count = 0$   
 $x = 1 \quad count = 1$   
 $x = 1.1 \quad count = 0$

⊗ Essendo la terza chiamata effettuata con un tipo  
double diverso da quello prima il count ritorna a zero.

14) Quale è il seguente output del programma?

```
#include <iostream>
using namespace std;
class Abe {
public:
    void f()
    void g()
    long double x; // è un long double
};

int main() {
    cout << sizeof(Abe) << endl;
}
```

- A) 16  $\neq$  long double = 16
- B) 4
- C) 2
- D) errore di compilazione

15) Cosa fa la funzione boo()

```
int Lista::boo() {
    Nodo * aux = testa;
    int c = 0, i = 0;
    while (aux)
        if (aux->valore == 10)
            i++;
    aux = aux->succ
}
```

return c;  $\rightarrow$  C è il numero di elementi trovati in Lista

(B) Restituisce il numero di elementi nella lista.

16) Non si esegue

17) Considerando il seguente codice, quale delle chiamate della funzione `max()` genera un errore di compilazione

#include <iostream>

using namespace std;

```
template <typename T>
T max (T x, T y) {
    return (x > y) ? x : y;
}
```

```
int main () {
    cout << max (3, 7) << endl;
    cout << max (3.0, 7.0) << endl;
    cout << max (3, 7.0) << endl;
    cout << max <float> (3, 7.0);
    return 0;
}
```

}

A) `max (3, 7);`

B) `max (3.0, 7.0);`

C) `max (3, 7.0);`  $\otimes$  due tipi diversi per uno stesso Template.

D) `max <float> (3, 7.0);`

18) Cosa fa la funzione `bool()`,

`int boo(Lista < T > & testa, T & x, int i = 20)` {

`Lista < T > q = testa, i = 0;`

`while (q != NULL) {`

~~`if (x == q->getValore()) i++;`~~

`q = q->succ;`

`}`

`return i;`

`}`

c) Conta il numero di elementi uguali ad 'x' presenti nella lista puntata da Testa.

19)

`int x, y;`

`int * const p = &x;`

`const int * p = &y;`

`q = &x;`

`*y = 4;` → genera errore

20) Il collegamento dinamico è lo strumento utilizzato per la realizzazione.

A) della memoria dinamica,

B) dell'ereditarietà,

C) del polimorfismo, 

D) dalle scomposizioni di classi;

21) linea 108 nella funzione inverti()

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

// Abbiamo QVI

}

A) testa = prev;

B) testa = current;

C) testa = next;

D) testa = nullptr;

22) Nel caso peggiore, qual è il numero di confronti necessari per la ricerca di un elemento all'interno di una lista concatenata semplice?

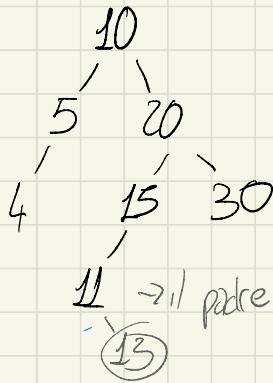
A)  $\log n$

B)  $\frac{n}{2}$

C)  $\log(n-1)$

D)  $O(n)$

23) Consideriamo il seguente BST. Se effettassimo l'inserimento di un nuovo nodo con chiave 13, che valore avrebbe la chiave del suo nodo padre?



B) 11 è il padre

24) Considerando la seguente sequenza di chiavi inserite in un BST inizialmente vuoto

Quali saranno i due figli del nodo con chiave 10 dopo aver eliminato il nodo con chiave 5?

NON SI POTESSE  
A leggera

25) Quale è l'output del seguente frammento di codice?

```
NodeBST<T> *ptr = root;  
while(ptr->getParent() != NULL)  
    cout << ptr->getKey() << endl;
```

3  
while (ptr)

```
cout << ptr->getKey() << endl;  
ptr = ptr->getRight();
```

3

A) Stampa ricorsivamente tutti i figli destri a partire dalla radice.  $\otimes$

B) Stampa ricorsivamente la chiave del parent fino alla radice, poi stampa tutte le chiavi dei figli destri.

C) Questo frammento non può essere compilato.

D) Stampa la chiave della root all'infinito.

26) Considerando la seguente sequenza di chiavi inserite in un BST strettamente vuoto.

Quali sono i due figli del nodo con chiave 5?

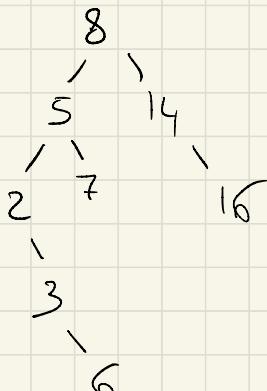
8 5 7 2 3 6 14 16

A) 2 e 6

B) 3 e 6

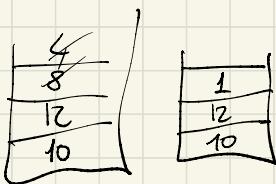
C) 2 e 7  $\otimes$

D) 3 e 7



28) Si considerino le seguenti operazioni in pila. Quanti `pop()` sono necessari per estrarre il numero 12?

```
push(10);  
push(12);  
push(8);  
push(4);  
pop();  
pop();  
push(1);
```



B) 2 Ø due pop

29) L'operazione di enqueue nell'implementazione con una lista linkata con il solo puntatore alla testa avviene in tempo

A) Log

B) Lineare Ø  $O(n)$

C) Quadratico

D) Costante

30) L'operazione di enqueue nell'implementazione con una lista linkata con puntatori alla testa e alla coda avviene in tempo

D) Ø(1) Costante

18. Qual è l'output del seguente programma?

```
#include <iostream>
using namespace std;
class abc {
    void f();
    void g();
    long double x;
};

int main() {
    cout << sizeof(abc) << endl;
}
```

- A) 16
- B) 4
- C) 2
- D) errore di compilazione

19. Qual è l'output del seguente codice?

```
int n = 100;
int a[n] = {0};
n = 200;
n += n;
cout << sizeof(a)/sizeof(int) << endl;
```

- A) 4
- B) 0
- C) 100
- D) 200

20. L'ultima riga delle seguenti funzione `Invert()` ha una istruzione mancante. Quale  
è questa istruzione? Dovevrebbe essere inserita alla fine della procedura per permettere  
che la lista sia data in input?

```
// questa funzione invierte la lista concatenata
void invert(Node* &testa) {
    Node* prev = NULL;
    Node* current = testa;
    Node* next;
    while (current != NULL) {
        next = current->succ;
        current->succ = prev;
        prev = current;
        current = next;
    }
}
```

//AGGIUNGERE UNA RIGA QUI

- A) testa = prev;
- B) testa = current;
- C) testa = next;
- D) testa=NULL;

20. Considerando la seguente sequenza di inserti inseriti in un BST malamente vuoto.

Quali sono i due figli del nodo con chiave 5?

8 5 7 2 3 6 14 16

- A) 2 e 6
- B) 2 e 7
- C) 3 e 7
- D) 3 e 8

30. Nel caso peggiore, quali è il numero di confronti necessari per la ricerca di un  
elemento all'interno di una lista completamente esplorata?

- A)  $\log n$
- B) 1
- C)  $n(n - 1)$
- D) 0

RISPOS

1 C 2.  
11 B 12.  
21 C 22.

$(\text{val}) < \text{Val} + 8$

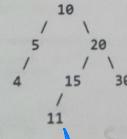
```
int func(Nodo *p) {
    return (p == nullptr) ||
           (p->succ == nullptr) ||
           (p->valore <> p->succ->valore
            && func(p->succ));
}
```

- A) Gli elementi della lista sono tutti diversi
- B) Gli elementi della lista sono ordinati in maniera non decrescita.
- C) Gli elementi della lista sono ordinati in maniera non crescente
- D) Nessuna delle precedenti

26. Quale struttura dati dinamica viene utilizzata dal gestore delle chiamate di funzioni nell'esecuzione di un programma?

- A) BST
- B) Lista concatenata
- C) Stack
- D) Coda

27. Considerando il seguente BST. Se effettuassimo l'inserimento di un nuovo nodo con chiave 13, che valore avrebbe la chiave del suo padre?



- A) 10
- B) 11
- C) 15
- D) 20

```

class Nodo {
    int valore;
    Nodo* succ;
    Nodo* prev;
};

};


```

\* A)   
 $X->prec->succ = X->succ;$   
 $X->succ->prec = X->prec;$

\* B)   
 $X->prec->succ = X->succ;$   
 $X->succ->prec = X->prec;$  NO

\* C)   
 $X->prec->succ = X->prec;$   
 $X->succ->prec = X->prec;$  NO

\* D)   
 $X->prec->succ = X->prec;$   
 $X->succ->prec = X->succ;$  NO

24. L'operazione di enqueue nell'implementazione con una lista linkata con il solo puntatore alla testa avviene in tempo

- A) Logaritmico -  $O(\log n)$
- B) Lineare -  $O(n)$
- C) Quadratico -  $O(n^2)$
- D) Costante -  $O(1)$

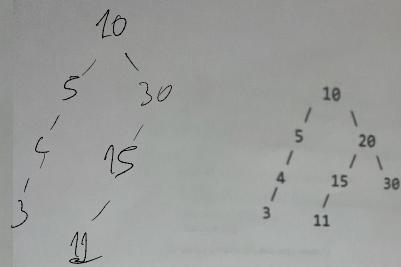
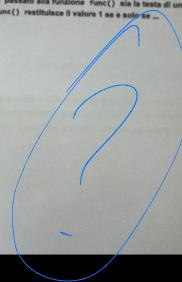
25. Supponendo che il parametro p passato alla funzione func() sia la testa di una lista concatenata, la funzione func() restituisce il valore 1 se e solo se ...

```

class Nodo {
    int valore;
    Nodo* succ;
};

};


```



- A) 3 4 5 10 11 15 30
- B) 10 5 4 3 20 15 11
- C) 3 4 5 11 15 30 10
- D) 3 11 4 15 20 10

22. Qual è l'output del seguente frammento di codice?

```

Nodo*BT11(*ptr = root;
while(*ptr->getParent()) {
    cout << ptr->getKey() << endl;
}

while(ptr) {
    cout << ptr->getKey() << endl;
    ptr = ptr->getRight();
}

```

- B stampa ricorrendo tutti i figli destra a partire dalla radice
- C stampa ricorrendo la chiave dei parent fino alla radice, poi stampa tutte le chiavi dei figli destra
- D Questo frammento non può essere compilato
- E Stampa la chiave della root all'infinito

23. Una lista doppiamente linkata è definita usando la classe nodo seguente, dove prec e succ rappresentano i puntatori agli elementi adiacenti. Quale del seguenti segmenti di codice deve essere eseguito per eliminare correttamente il nodo puntato da X, assumendo che X non punti ne all'inizio ne alla fine della lista ?

20. Qual è l'output del seguente codice ?

```
char str1[] = {"C","i","s","o"};
char str2[] = {"Ciao"};

cout << sizeof(str1) << endl;
cout << sizeof(str2) << endl;
```

- A)

4  
4

- B)

5  
- coda /n/  
4  
5  
 D)  
5  
4

21. Considerando il seguente BST. Indicare quale scrive la sequenza dei nodi visitati da una visita posteriore, dopo aver eliminato il nodo con chiave 20.

Vedi foto  
Alta Sinistra  
Pallina prima

- A) Restituisce il numero di volte che il valore 10 è presente nella lista puntata da "lista".
- B) Restituisce il numero di elementi presenti nella lista puntata da "lista".
- C) Restituisce zero in ogni caso.
- D) Scorre tutta la lista puntata da "lista" stampando le sue chiavi solo se sono uguali a 10.

12. Dalla seguente classe, accorgersi la definizione corretta per la funzione membro f

```
template <class T>
class abc {
    void f();
};
```

X

```
template <class T> void abc<T>::f() {}
```

A)

```
template <class T> void abc::f() {}
```

C)

```
template <T> void abc<class T>::f() {}
```

D)

```
template <T> void abc<T>::f() {}
```

13. Una delle seguenti affermazioni è vera per una funzione inline

- A) Viene eseguita più rapidamente poiché trattata come una macro internamente
- B) Viene eseguita più rapidamente perché gli viene attribuita una priorità più alta rispetto a una funzione normale
- C) Non viene eseguita più velocemente delle altre funzioni
- D) nessuna delle precedenti

14. Quale riga genera errore di compilazione?

```
int x,y;
int* const p = &x;
const int *q = &y;
q = &x;
*q = 4;
```

E' errore

## Esercizi fisi:

Selezionare la corretta definizione di una funzione virtuale pura: `virtual void g() = 0;`

A) `virtual void f() = 0;`

B) `void virtual f() = 0;`

C) `virtual void f() = 0;` X

D) Nessuna delle precedenti.

2) L'operazione di enqueue nell'implementazione con una lista finita con puntatori alla testa e alla coda avviene in tempo:  $O(1)$  Costante.

A)  $O(\log n)$

B)  $O(n)$

C)  $O(n^2)$

D)  $O(1)$  X

3) Un header definito dall'utente viene incluso usando la sintassi....

A) `#include "file.h"` X

B) `#include <file.h>`

C) `# include <file>`

D) `#include file.h`

4) Quali di questi Algoritmi è basato sul paradigma Divide et Impera? Merge - Quick

A) InsertionSort - Selection

B) MergeSort e QuickSort

C) MergeSort e InsertionSort

D) QuickSort e SelectionSort

5) Nel caso peggiore, qual è il numero di confronti necessari per la ricerca di un elemento all'interno di una lista concatenata semplice?

A)  $O(\log n)$

B)  $O(\frac{n}{2})$

C)  $O(\log n-1)$

D)  $O(n)$

6) Caso intendiamo dire quando diciamo che un algoritmo  $X$  è assolutamente più efficiente di un altro algoritmo  $Y$ ?

A)  $X$  è sempre migliore di  $Y$  per piccoli input.

B)  $X$  è sempre migliore di  $Y$  per grandi input.

C)  $Y$  è sempre migliore di  $X$  per piccoli input.

D)  $Y$  è sempre migliore di  $X$  per grandi input.

7) Una classe Astratta è una classe che:

- A) Deve contenere tutte funzioni virtuali pure.
- B) Deve contenere almeno una funzione virtuale pura. 
- C) Può non contenere funzioni virtuali pure
- D) Deve contenere funzioni virtuali pure definite fuori dalla classe.

8) class A {

...

friend void foo

}

A) La funzione foo può accedere direttamente a tutti i membri di A per non essere membro



B) La funzione foo è un membro privato di A.

C) La funzione foo può essere richiamata con la notazione A.foo

D) La funzione foo può accedere solo ai membri pubblici di A

Q) Class A {

int a;

...

protected:

char b;

...

public:

float c;

...

}

Class B: private A {

...

}

A) La classe B eredita tutti i membri di A.

B) B eredita tutti i membri private di A.

C) La classe B eredita i membri protected e private di A e li rende public.

D) La classe B eredita i membri protected e public di A e li rende public.  
Private  $\otimes$

10) L'implementazione di una pila con Array statico:

A) Trasforma la pila in una coda

B) Limita la dimensione Massima della Pila  $\otimes$

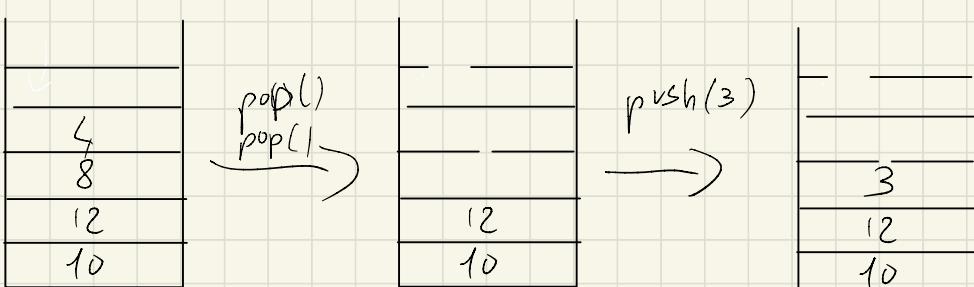
C) Consente di effettuare tutte le operazioni in tempo Costante.

D) Non è possibile.

Si considerino le seguenti operazioni su una pila:

```
push(10)
push(12)
push(8)
push(4)
pop()
pop()
push(3)
```

Quanti pop() sono necessari per estrarre il numero 12?



Per estrarre il 12 Servono 2 pop()

#### Quesito

Supponendo che il parametro p passato alla funzione func() sia la testa di una lista concatenata, la funzione func() restituisce il valore 1 se ...

```
class Nodo{
public:
    int valore;
    Nodo *succ;
};

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

Supponiamo di avere i puntatori al primo e all'ultimo elemento di

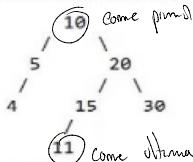
Gli elementi della lista sono in modo NON  
Decrescente.

}

Supponiamo di avere i puntatori al primo e all'ultimo elemento di una lista concatenata semplice. Quale di queste operazioni ha un costo che dipende dalla lunghezza della lista?

Eliminazione ultimo elemento  
della lista.

Indicare quale sequenza di input delle chiavi genera il seguente BST



- A) 10 20 5 11 15 4 30
- B) 10 5 20 20 11 15 4
- C) 10 5 20 4 30 15 11 X
- D) Nessuna delle precedenti

Una lista deve essere bidirezionale perché è definita secondo la chiave risulta

```

void func(int n)
{
    int count = 0;
    for (int i=n/2; i<=n; i++)
        for (int j=1; j<=i; j=i*2)
            for (int k=1; k<=n; k=k*2)
                count++;
}

```

$O(n \log n)$

```

void func(int n)
{
    int count = 0;
    for (int i=n/2; i<=n; i++)
        for (int j=1; j<=i; j++)
            for (int k=1; k<=n; k=k*2)
                count++;
}

```

$O(n^2 \log n)$

```

void func(int n)
{
    int count = 0;
    for (int i=0; i<n; i++)
        for (int j=0; j<i; j++)
            if (j>n == 0)
                {
                    for (int k=0; k<j; k++)
                        count<<=2;
                }
}

```

(?)  $O(n^5)$  perché?

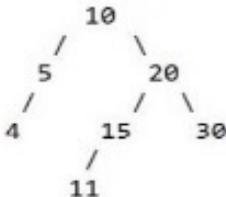
```

((p->valore >= p->succ->valore
  && func(p->succ))
);

```

Supponiamo di avere i puntatori al primo e all'ultimo elemento di una lista concatenata semplice. Quale di queste operazioni ha un costo che dipende dalla lunghezza della lista?

Indicare quale sequenza di input delle chiavi genera il seguente BST



Una lista doppiamente linkata è definita usando la classe nodo seguente, dove prec e succ rappresentano i puntatori agli elementi adiacenti. Quale dei seguenti segmenti di codice deve essere eseguito per eliminare correttamente il nodo puntato da X, assumendo che X non punti né all'inizio né alla fine della lista?

```

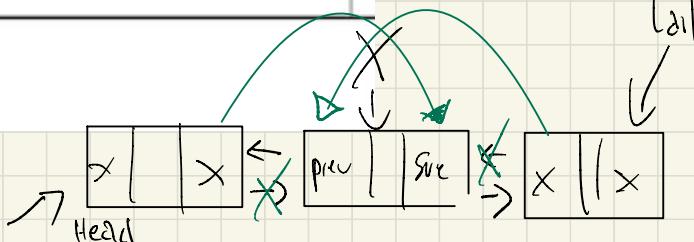
Nodo{
public:
    int valore;
    Nodo *succ;
    Nodo *prec;
};

```

Gia fatto! Eliminazione ultimo elemento

Gia fatto

$X \rightarrow buee \rightarrow prev = X \rightarrow prev$ .  
 $X \rightarrow prev \rightarrow Suce = X \leftarrow Suce$ .



Ogni algoritmo di ordinamento basato su confronti richiede almeno

...

Scegliere l'utilizzo corretto di **delete** per l'espressione `ptr = new int[100]`

L'accesso di default ad un membro di classe è

Qual è l'output del seguente frammento di codice?

```
NodeBST<T> *ptr = root;  
while(ptr->getParent() != NULL) {  
    cout << ptr->getKey() << endl;  
}  
while(ptr) {  
    cout << ptr->getKey() << endl;  
    ptr = ptr->getRight();  
}
```

$O(\frac{n}{2})$

$\rightarrow \text{delete}[\ ] \text{ptr};$   
 $\rightarrow \text{private}$

i figli destri del BST stampati ricorsivamente

Scegliere la corretta definizione di una funzione virtuale pura

$\rightarrow \text{virtual void g()=0;}$

L'operazione di enqueue nell'implementazione con una lista linkata con puntatori alla testa e alla coda avviene in tempo

$\rightarrow O(1)$

Un header definito dall'utente viene incluso usando la sintassi ...

$\rightarrow \#define "file.h"$

Quali di questi algoritmi di ordinamento sono basati sul paradigma divide et impera?

$\rightarrow \text{Merge - Quick}$

Nel caso peggiore, qual è il numero di confronti necessari per la ricerca di un elemento all'interno di una lista concatenata semplice?

$\rightarrow O(n)$

Cosa intendiamo dire quando diciamo che un algoritmo X è asintoticamente più efficiente di un altro algoritmo Y?

$\rightarrow X$  è migliore di  $Y$  anche per grandi Input

T1 =  $5n^2 + 2n - 10$  T2 =  $5\sqrt{n} + 22$  T1 > T2

$\rightarrow n^2 + c$

T1 =  $5\sqrt{bn}^3$  T2 =  $5\sqrt{cn}^5$  T1 > T2

$5\sqrt{b}n^3$        $5\sqrt{c}n^5$        $25n(n^3 \cdot n^5) = 25n(n^8) \rightarrow (n^8)$

T1 =  $O(n \log^2 n)$  T2 =  $O(1.5n \log n)$

$\rightarrow T_1 > T_2$

T1 =  $O((5^n)n \log n)$  T2 =  $O((4^n)n)$   $O(5^n) n \log n \mid O(4^n) n \rightarrow T_1 > T_2$

f(n)=O(g(n)) se c,n>0

T1 =  $5n^2 + 2n - 10$  T2 =  $5\sqrt{n} + 22$  T1 > T2       $5n^2 - 2n \mid 5\sqrt{n} \quad 23(n^2, n^2) - 2n = 23(n^3) - 2n = \frac{23}{n^2}$

T1 =  $n^e$  T2 =  $n^k$  G =  $\log T_1/T_2$

$n^e \quad n^k \quad \log \frac{n^e}{n^k} = \log(n^{e-k}) (e-k)(\log(n)) = \log(n)$

n var c>1

$O(n) \leq O(\log n) < O(n^e)$

Selection Sort

int x, y;  $\rightarrow$  int\* const p = &x;  $\rightarrow$  const int\* q = &y;  $\rightarrow$  q = &x;  
 $\rightarrow$  \*q = 4; trova l'errata

$*q = 4$  errore

```
#include <iostream>
using namespace std;
class abc {
public:
    static int x;
    int i;
    abc() { i = ++x; }
};
int abc::x;
main(){
    abc m, n, p;
    cout << m.x << " " << n.x;
}
```

(No  
Indicazioni)

Dalla seguente classe, scegliere la definizione corretta per la funzione membro f

```
template <class T>
class abc {
    void f();
};
```

Trovare le complessità dei seguenti algoritmi

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
    for (j = 2; j <= n; j = j * 2) {
        k = k + n / 2;
    }
}
```

template <class T> void abc<T>::f()

$O(n \log n)$

```
int a = 0, i = N;
while (i > 0) {
    a += i;
    i /= 2;
}
```

Divisione

$O(\log n)$

### Quesito

Considerando la seguente semplice implementazione di una lista concatenata, cosa effettua la funzione boo()?

```
class Nodo{
public:
    int valore;
    Nodo *succ;};
class ListaConcatenata{
public:
    Nodo* testa;};
void boo(Nodo* testa){
if(testa == nullptr) return;
cout << testa->valore << endl;
boo(testa->succ);
}
```

Stampa ricorsivamente i valori della lista

L'ultima riga della seguente funzione inverti() ha una istruzione mancante. Quale di queste istruzioni dovrebbe essere inserita alla fine della procedura per permettere alla funzione inverti() di invertire l'ordine degli elementi della lista concatenata la cui testa è data in input?

```
/* 'testa' punta alla testa di una lista concatenata */
void inverti(Nodo* testa) {
    Nodo* prev = nullptr;
    Nodo* current = *testa;
    Nodo* next;
    while (current != nullptr) {
        next = current->succ;
        current->succ = prev;
        prev = current;
        current = next;
    }
}
```

Head = prev;

