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

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

class ListaConcatenata{
  public:
     Nodo* testa;
};

void boo(Nodo* testa){
  if(testa==nullptr) return;
  boo(testa->succ);

  cout<<testa->valore<<endl;
}
```

1. Se testa è un puntatore nullo non fa niente, altrimenti stampa il valore che contiene

2. Se testa corrisponde alla testa di una lista concatenata, stampa tutto il contenuto della lista dalla testa alla coda.

3. se testa corrisponde alla testa di una lista concatenata, stampa tutto il contenuto della lista in ordine inverso // risposta esatta

4. Il codice contiene un errore e non può essere compilato.

2. 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 dalla
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;
  }
  //AGGIUNGERE RIGA QUI
}
```

1. testa = prev; //risposta esatta
2. testa = current;
3. testa = next;
4. testa = nullptr;

3. 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

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

1. Gli elementi della lista sono tutti diversi
2. Gli elementi della lista sono ordinati in maniera non decrescente // risposta esatta
3. Gli elementi della lista sono ordinati in maniera non crescente
4. Nessuna delle precedenti

4. 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?

1. Eliminare il primo elemento
2. Inserire un nuovo elemento in testa
3. Eliminare l'ultimo elemento della lista
4. Aggiungere un nuovo elemento alla fine della lista // risposta esatta

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

1. log n
2. n/2
3. log(n-1)
4. n // risposta esatta

6. Qual è il numero minimo di campi per ciascun nodo di una lista doppiamente linkata?

1. 1
2. 2
3. 3 // risposta esatta prev/valore/succ
4. 4

□

7. 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;
};
```

1. X->prec->succ = X->succ; X->succ->prec = X->prec; // risposta esatta
2. X->prec.succ = X->succ; X.succ->prec = X->prec;
3. X.prec->succ = X.prec; X->succ.prec = X.prec;

4. X->prec->succ = X->prec; X->succ->prec = X->succ;

8. Quale di queste strutture dati implementa una politica LIFO?

1. BST
2. Coda
3. Stack // risposta esatta
4. Lista

9. L'implementazione di una pila con un array statico

1. Trasforma la pila in una coda
2. Limita la dimensione massima della pila // risposta esatta
3. Consente di effettuale tutte le operazioni in tempo costante
4. Non è possibile

10. Qual è la complessità di questa funzione?

```
func(int n)
{
  if(n==1) return;
  for(int i=1;i<=n;i++)
  {
     for(int j=1;j<=n;j++)
     {
        cout<<"@@@";
        break;
     }
  }
}
```

□

1. O(N) // risposta esatta dovrebbe essere O(n alla seconda)
2. O(sqrt(N))
3. O(N/2)
4. O(logN)

11. Qual è la complessità di questa funzione?

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

1. O(n^2)
2. O(n log^2 n) // risposta
3. O(n log n)
4. O(log n)

12. Come eredita i membri public e protected della classe A?

```
Class A
{
      Int a;

      protected:
            Char b:
            …
      public:
            float c;
            …
```

};

Class B : private A
{
    …
};

Risposta: La classe B li eredita rendendoli entrambi privati

13. Una classe astratta è una classe che: …

Risposta: Contiene una funzione virtuale pura.

14. Ogni algoritmo di ordinamento basato su confronti richiede:

a) n*log(n) passi **b)** n^2 //risposta esatta **c)** log(n) passi **d)** Nessuna delle precedenti

15. Cosa fa il seguente codice?

```
2   NodeBST<T> *ptr = root;
3
4   while(ptr->getParent())
5       cout << ptr->getKEy() << endl;
6
7   while(ptr)
8   {
9       cout << ptr->getKey() << endl;
10      ptr = ptr->getRight();
11  }
```

a) Stampa ricorsivamente tutti i figli destri a partire dalla radice. // risposta esatta

b) Stampa ricorsivamente la chiave del parent fino alla radice,

poi stampa tutte le chiavi del figlio.

c) Questo frammento non puo essere compilato.

d) Stampa le chiavi della root all'infinito.

Risposta giusta: **a)**

16. Qual è la complessità di questa funzione?

```
1  int i, j, k = 0;
2
3  for(i = n/2; i <= n; i++)
4      for(j = 2; j <= n; j = j*2)
5          k = k + n/2;
```

a) $O(n^2)$
b) O(n*log(n)) // risposta esatta
c) O(n/2)
d) O(log(n)/2)

17. Qual è la complessità di questa funzione?

```
1  int a = 0, i = N;
2
3  while(i > 0)
4  {
5      a += i;
6      i /= 2;
7  }
```

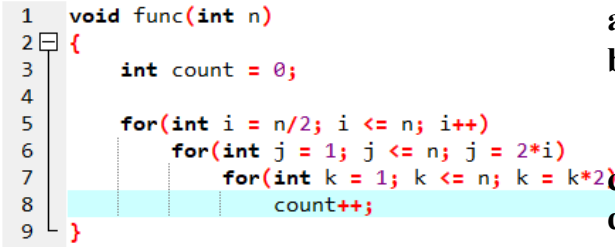
a) O(n)
b) O(
c) O(n/2)
d) O(log(n)) // risposta esatta

18. Qual è la complessità di questa funzione?

```
1  func(int n)
2  {
3      if(n == 1)
4          return;
5
6      for(int i = 1; i <= n; i++)
7          {
8              for(int j = 1; j <= n; j++)
9                  {
10                     cout << " ###";
11                     break;
12                 }
13         }
14 }
```

a) O(n) // risposta esatta
b) O(
c) O(n/2)
d) O(log(n))

19. Qual è la complessità di questa funzione?



```
1  void func(int n)
2  {
3      int count = 0;
4
5      for(int i = n/2; i <= n; i++)
6          for(int j = 1; j <= n; j = 2*i)
7              for(int k = 1; k <= n; k = k*2)
8                  count++;
9  }
```

a) $O(n^2)$
b) $O(n * \log^2(n))$ // risposta esatta
c) $O(n * \log(n))$
d) $O(\log(n))$