```
g++ -c -Wall -g -std=c++11 -static-libstdc++ Nome.cpp -o Nome.o
g++ -Wall -g -std=c++11 -static-libstdc++ Nome.o -o Nome.exe
```

ESERCIZIO 01

Creare un array int di 10 elementi (nella stack) con valori randomici tra 1 e 30 ed eseguire un ciclo che trova i valori di massimo e di minimo. Stampare a schermo l'array e questi due valori.

ESERCIZIO 02

Creare un array int di 10 elementi (nella stack) con valori randomici tra 1 e 30 e trovare i suoi valori di massimo e di minimo utilizzando una funzione che prende in input un puntatore e la dimensione dell'array. Stampare a schermo l'array e questi due valori.

ESERCIZIO 03

Creare un array int di N elementi (dove N è un valore inserito da terminale dall'utente) (nella heap) con valori rndomici tra 1 e 30 e utilizzare una funzione (che prende in input un puntatore e la dimensione dell'array) per stampare a schermo i valori dell'array e la quantità di numeri pari in esso contenuti.

ESERCIZIO 04

Creare una funzione swap() cui dati in input due puntatori scambia tra di loro il valore delle variabili cui puntano.
Usare questa funzione all'interno del main() (scegliete voi i numeri di esempio)

ESERCIZIO 05

Creare un namespace di nome pog contenente una variabile intera "finoA" e una funzione "numeriPrimi()" che calcola tutti i numeri primi fino a "finoA"
Fuori dal namespace programmare un main() che fa inserire il valore di finoA all'utente e poi richiama la funzione numeriPrimi().

ESERCIZIO 06

Creare un array int di N elementi (dove N è un valore inserito da terminale dall'utente) (nella heap) con valori incrementali (il primo elemento vale 0, il secondo 1, il terzo 2..) .
Creare un metodo (che prende in input un puntatore e la dimensione dell'array) per stampare a schermo i valori dell'array e la quantità di numeri pari in esso contenuti.
Creare un metodo che prende in input un puntatore, la dimensione dell'array e un altro intero che dovra' essere la nuova dimensione dell'array. Questo metodo dovra' restituire l'array ingrandito (o rimpicciolito), se vengono aggiunti nuovi elementi i loro valori dovranno anch'essi essere incrementali, ma l'incremento dovrà essere pari a 2 (ad esempio se l'array iniziale fosse stato di tre elementi con a[2] = 2, il quarto elemento dovrà valere a[3] = 4)
Nel main generare il primo array, stamparlo, cambiarne la dimensione (prendendo in input dall'utente la nuova dimensione) e stamparlo di nuovo.

ESERCIZIO 07 Esercizio sulle liste.
Una lista non è altro che un insieme di oggetti (di una qualche class o struct vostra) che tra le variabili membro hanno un puntatore ad un altro oggetto del suo stesso tipo. Questo puntatore solitamente si chiama next e punta all'oggetto successivo all'interno della lista.
Per scorrere una lista quindi basta sapere l'indirizzo del primo elemento, da questo elemento si può trovare il successivo utilizzando il puntatore next, da questo nuovo elemento si potrà trovare il successivo ancora utilizzando questo nuovo next, e così via fino a trovare un next nullo, il che vorrà dire che la lista è terminata.

Creare una struttura Stage che contenga i campi interi id e difficulty e un puntatore ad un oggetto di tipo Stage di nome next.
La struttura dovrà avere un costruttore che prende in input i valori da assegnare a id e difficulty e settare next a nullptr.

Nel main instanziare uno Stage (nella heap) con id pari a -1 e difficulty 0 assegnandolo ad un puntatore di nome head.
Dichiarare poi un puntatore ad un oggetto di tipo Stage (non assegnategli alcun valore per ora) di nome iter.
Inserire un loop infinito che chiede all'utente di inserire un char per selezionare una scelta tra:
a ---> "add a new stage"
r ---> "remove a stage"
q ---> "quit"
default ---> (non fa nulla)

Nel caso (a) si richiede all'utente di inserire due interi, il primo per l'id e il secondo per la difficulty da assegnare al nuovo Stage. Bisognerà dunque scorrere tutta la lista con un ciclo (sfruttando il puntatore iter) e una volta arrivati alla fine aggiungere un nuovo elemento (assegnando il suo indirizzo al puntatore dell'ormai ex-ultimo elemento).

Nel caso (r) si richiede all'utente di inserire un intero indicante l'indice dell'elemento della lista da rimuovere. Bisognerà dunque scorrere tutta la lista con un ciclo (sfruttando il puntatore iter) e una volta arrivati all'indice interessato (ammesso e non concesso che sia possibile, se la lista è più corta stampare a schermo un messaggio "Elemento non trovato") ed eliminare tale elemento (assegnando il suo indirizzo al puntatore dell'ormai ex-ultimo elemento). Nel caso in cui l'indice sia zero, bisognerà eliminare l'elemento cui punta head. Ovviamente occorre prestare molta attenzione, se si elimina l'head, head dovrà puntare al "nuovo" primo elemento. Se si elimina l'elemento N, l'elemento N-1 dovrà puntare all'elemento N+1 !

Nel caso (q) bisognerà eliminare tutti gli elementi della lista.
Sia chiaro che quando scrivo "eliminare" intendo non solo rimuoverli dalla lista ma anche liberare la memoria con il delete!