Esame 20220729

Esercizio 3

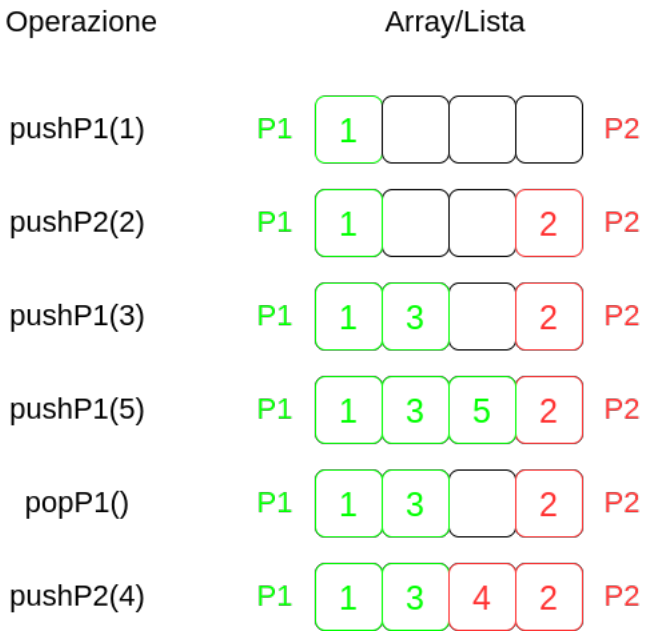
(1) **Esercizio 3 v1**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere nel file `pila.cc` l'implementazione di due pile di interi, chiamate P1 e P2, in modo tale che P1 e P2 condividano lo stesso array **oppure** la stessa lista doppiamente concatenata (lo studente sceglie se implementare P1 e P2 usando un'array oppure una lista doppiamente concatenata). In altre parole, P1 e P2 sono (e devono comportarsi come) due pile indipendenti tranne che per la memoria in cui salvano i loro elementi, che é condivisa. Di conseguenza, la dimensione massima di P1 e P2 é pari alla dimensione della memoria condivisa, ma la somma delle dimensioni di P1 e P2 non puó mai essere maggiore della memoria condivisa. Il file `pila.cc` deve contenere le seguenti funzioni (se presente, il valore di ritorno di una funzione é "true" se l'operazione é andata a buon fine, "false" altrimenti):

- `void init(int dim)`: inizializza P1 e P2 con un'array o una lista doppiamente concatenata di dimensione massima `dim` (allocazione dinamica). Sia P1 che P2 possono crescere fino a `dim`, con l'unica limitazione che la somma del numero di elementi in P1 e P2 non puó essere maggiore di `dim`;
- `bool pushP1(int valore)`: inserisci l'elemento 'valore' in P1;
- `bool pushP2(int valore)`: inserisci l'elemento 'valore' in P2;
- `bool topP1(int&)`: assegna al parametro il valore del primo elemento di P1;
- `bool topP2(int&)`: assegna al parametro il valore del primo elemento di P2;
- `bool popP1()`: rimuovi il primo elemento di P1;
- `bool popP2()`: rimuovi il primo elemento di P2;
- `void deinit()`: de-inizializza sia P1 che P2 e dealloca la memoria dinamica;
- `void print()`: stampa a video gli elementi di P1 e P2.

Questo é un esempio di esecuzione con a lato una rappresentazione grafica della pila:



Note:

- Creare un file dal nome `pila.cc` e scrivere dentro al file l'implementazione delle funzioni come da consegna. E' possibile scaricare i file `esercizio3.cc` (che contiene un main di prova) e `pila.h` (che contiene la definizione delle funzioni da implementare) per testare il codice scritto. Infine, caricare **solo** il file `pila.cc` nello spazio apposito;
- Non modificare i file `esercizio3.cc` e `pila.h`. In altre parole, l'implementazione delle funzioni deve essere compatibile con il codice scritto in `esercizio3.cc` e `pila.h`, poichè questi due file verranno usati per valutare la correttezza dell'esercizio;
- All'interno del file `pila.cc` **non é ammesso** l'utilizzo di funzioni di libreria al di fuori di quelle definite in `iostream`;
- All'interno del file `pila.cc` **é ammesso** l'utilizzo di variabili di tipo `static`, che possono essere usate per mantenere informazioni sullo stato delle pile;
- Ricordarsi di deallocare la memoria allocata dinamicamente;
- Ricordarsi di distinguere gli esempi nella descrizione dell'esercizio (che servono solo ad aiutare a comprendere il problema) dalle istruzioni di implementazione;
- Per essere considerata unica, una lista doppiamente concatenata deve permettere, a partire da un **qualsiasi** nodo, di visitare ogni altro nodo della lista in maniera contigua.

Information for graders:

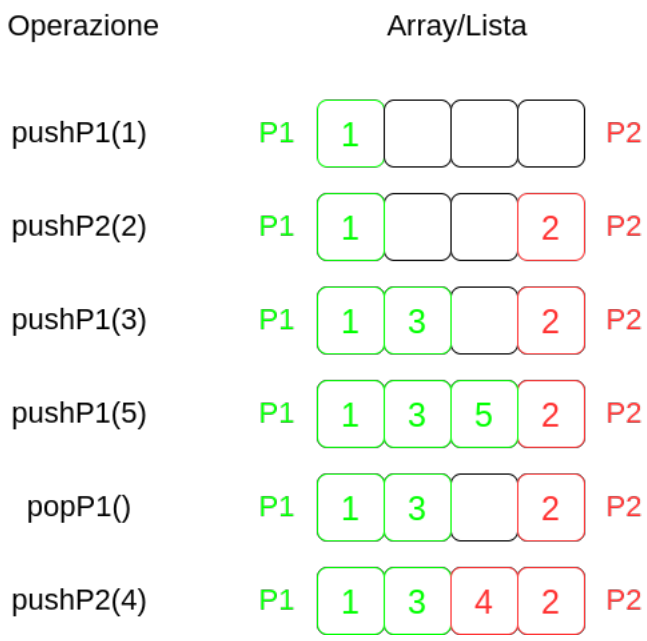
(2) **Esercizio 3 v2**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere nel file `pila.cc` l'implementazione di due pile di float, chiamate P1 e P2, in modo tale che P1 e P2 condividano lo stesso array **oppure** la stessa lista doppiamente concatenata (lo studente sceglie se implementare P1 e P2 usando un'array oppure una lista doppiamente concatenata). In altre parole, P1 e P2 sono (e devono comportarsi come) due pile indipendenti tranne che per la memoria in cui salvano i loro elementi, che é condivisa. Di conseguenza, la dimensione massima di P1 e P2 é pari alla dimensione della memoria condivisa, ma la somma delle dimensioni di P1 e P2 non puó mai essere maggiore della memoria condivisa. Il file `pila.cc` deve contenere le seguenti funzioni (se presente, il valore di ritorno di una funzione é "true" se l'operazione é andata a buon fine, "false" altrimenti):

- `void init(int dim)`: inizializza P1 e P2 con un'array o una lista doppiamente concatenata di dimensione massima `dim` (allocazione dinamica). Sia P1 che P2 possono crescere fino a `dim`, con l'unica limitazione che la somma del numero di elementi in P1 e P2 non puó essere maggiore di `dim`;
- `bool pushP1(float valore)`: inserisci l'elemento 'valore' in P1;
- `bool pushP2(float valore)`: inserisci l'elemento 'valore' in P2;
- `bool topP1(float&)`: assegna al parametro il valore del primo elemento di P1;
- `bool topP2(float&)`: assegna al parametro il valore del primo elemento di P2;
- `bool popP1()`: rimuovi il primo elemento di P1;
- `bool popP2()`: rimuovi il primo elemento di P2;
- `void deinit()`: de-inizializza sia P1 che P2 e dealloca la memoria dinamica;
- `void print()`: stampa a video gli elementi di P1 e P2.

Questo é un esempio di esecuzione con a lato una rappresentazione grafica della pila:



Note:

- Creare un file dal nome `pila.cc` e scrivere dentro al file l'implementazione delle funzioni come da consegna. E' possibile scaricare i file `esercizio3.cc` (che contiene un main di prova) e `pila.h` (che contiene la definizione delle funzioni da implementare) per testare il codice scritto. Infine, caricare **solo** il file `pila.cc` nello spazio apposito;
- Non modificare i file `esercizio3.cc` e `pila.h`. In altre parole, l'implementazione delle funzioni deve essere compatibile con il codice scritto in `esercizio3.cc` e `pila.h`, poichè questi due file verranno usati per valutare la correttezza dell'esercizio;
- All'interno del file `pila.cc` **non é ammesso** l'utilizzo di funzioni di libreria al di fuori di quelle definite in `iostream`;
- All'interno del file `pila.cc` **é ammesso** l'utilizzo di variabili di tipo `static`, che possono essere usate per mantenere informazioni sullo stato delle pile;
- Ricordarsi di deallocare la memoria allocata dinamicamente;
- Ricordarsi di distinguere gli esempi nella descrizione dell'esercizio (che servono solo ad aiutare a comprendere il problema) dalle istruzioni di implementazione;
- Per essere considerata unica, una lista doppiamente concatenata deve permettere, a partire da un **qualsiasi** nodo, di visitare ogni altro nodo della lista in maniera contigua.

Information for graders:

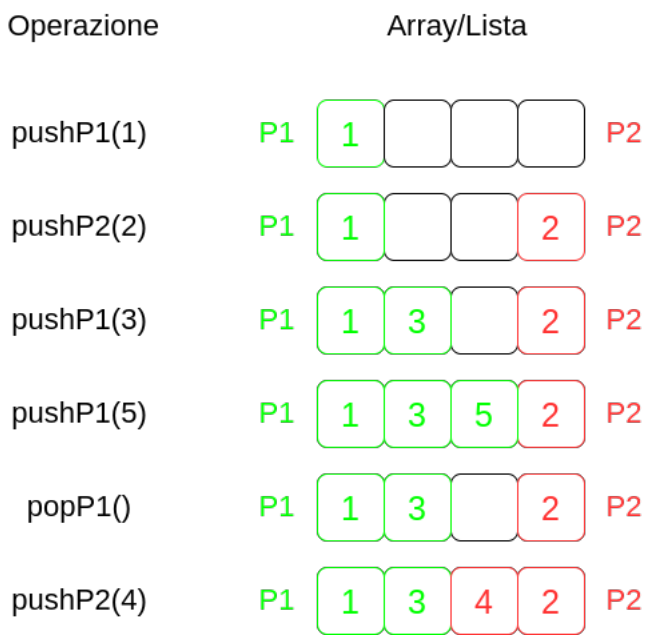
(3) **Esercizio 3 v3**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere nel file `pila.cc` l'implementazione di due pile di double, chiamate P1 e P2, in modo tale che P1 e P2 condividano lo stesso array **oppure** la stessa lista doppiamente concatenata (lo studente sceglie se implementare P1 e P2 usando un'array oppure una lista doppiamente concatenata). In altre parole, P1 e P2 sono (e devono comportarsi come) due pile indipendenti tranne che per la memoria in cui salvano i loro elementi, che é condivisa. Di conseguenza, la dimensione massima di P1 e P2 é pari alla dimensione della memoria condivisa, ma la somma delle dimensioni di P1 e P2 non puó mai essere maggiore della memoria condivisa. Il file `pila.cc` deve contenere le seguenti funzioni (se presente, il valore di ritorno di una funzione é "true" se l'operazione é andata a buon fine, "false" altrimenti):

- `void init(int dim)`: inizializza P1 e P2 con un'array o una lista doppiamente concatenata di dimensione massima `dim` (allocazione dinamica). Sia P1 che P2 possono crescere fino a `dim`, con l'unica limitazione che la somma del numero di elementi in P1 e P2 non puó essere maggiore di `dim`;
- `bool pushP1(double valore)`: inserisci l'elemento 'valore' in P1;
- `bool pushP2(double valore)`: inserisci l'elemento 'valore' in P2;
- `bool topP1(double&)`: assegna al parametro il valore del primo elemento di P1;
- `bool topP2(double&)`: assegna al parametro il valore del primo elemento di P2;
- `bool popP1()`: rimuovi il primo elemento di P1;
- `bool popP2()`: rimuovi il primo elemento di P2;
- `void deinit()`: de-inizializza sia P1 che P2 e dealloca la memoria dinamica;
- `void print()`: stampa a video gli elementi di P1 e P2.

Questo é un esempio di esecuzione con a lato una rappresentazione grafica della pila:



Note:

- Creare un file dal nome `pila.cc` e scrivere dentro al file l'implementazione delle funzioni come da consegna. E' possibile scaricare i file `esercizio3.cc` (che contiene un main di prova) e `pila.h` (che contiene la definizione delle funzioni da implementare) per testare il codice scritto. Infine, caricare **solo** il file `pila.cc` nello spazio apposito;
- Non modificare i file `esercizio3.cc` e `pila.h`. In altre parole, l'implementazione delle funzioni deve essere compatibile con il codice scritto in `esercizio3.cc` e `pila.h`, poichè questi due file verranno usati per valutare la correttezza dell'esercizio;
- All'interno del file `pila.cc` **non é ammesso** l'utilizzo di funzioni di libreria al di fuori di quelle definite in `iostream`;
- All'interno del file `pila.cc` **é ammesso** l'utilizzo di variabili di tipo `static`, che possono essere usate per mantenere informazioni sullo stato delle pile;
- Ricordarsi di deallocare la memoria allocata dinamicamente;
- Ricordarsi di distinguere gli esempi nella descrizione dell'esercizio (che servono solo ad aiutare a comprendere il problema) dalle istruzioni di implementazione;
- Per essere considerata unica, una lista doppiamente concatenata deve permettere, a partire da un **qualsiasi** nodo, di visitare ogni altro nodo della lista in maniera contigua.

Information for graders:

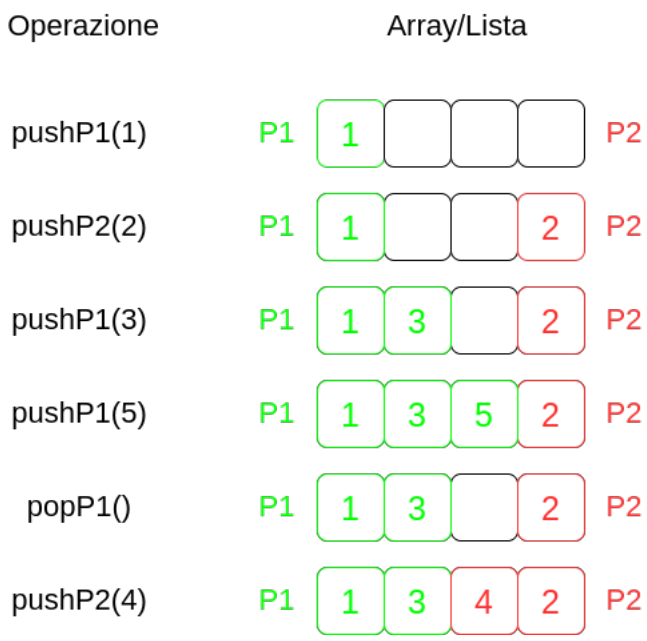
(4) **Esercizio 3 v3**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere nel file `pila.cc` l'implementazione di due pile di caratteri, chiamate P1 e P2, in modo tale che P1 e P2 condividano lo stesso array **oppure** la stessa lista doppiamente concatenata (lo studente sceglie se implementare P1 e P2 usando un'array oppure una lista doppiamente concatenata). In altre parole, P1 e P2 sono (e devono comportarsi come) due pile indipendenti tranne che per la memoria in cui salvano i loro elementi, che é condivisa. Di conseguenza, la dimensione massima di P1 e P2 é pari alla dimensione della memoria condivisa, ma la somma delle dimensioni di P1 e P2 non puó mai essere maggiore della memoria condivisa. Il file `pila.cc` deve contenere le seguenti funzioni (se presente, il valore di ritorno di una funzione é "true" se l'operazione é andata a buon fine, "false" altrimenti):

- `void init(int dim)`: inizializza P1 e P2 con un'array o una lista doppiamente concatenata di dimensione massima `dim` (allocazione dinamica). Sia P1 che P2 possono crescere fino a `dim`, con l'unica limitazione che la somma del numero di elementi in P1 e P2 non puó essere maggiore di `dim`;
- `bool pushP1(char valore)`: inserisci l'elemento 'valore' in P1;
- `bool pushP2(char valore)`: inserisci l'elemento 'valore' in P2;
- `bool topP1(char&)`: assegna al parametro il valore del primo elemento di P1;
- `bool topP2(char&)`: assegna al parametro il valore del primo elemento di P2;
- `bool popP1()`: rimuovi il primo elemento di P1;
- `bool popP2()`: rimuovi il primo elemento di P2;
- `void deinit()`: de-inizializza sia P1 che P2 e dealloca la memoria dinamica;
- `void print()`: stampa a video gli elementi di P1 e P2.

Questo é un esempio di esecuzione con a lato una rappresentazione grafica della pila:



Note:

- Creare un file dal nome `pila.cc` e scrivere dentro al file l'implementazione delle funzioni come da consegna. E' possibile scaricare i file `esercizio3.cc` (che contiene un main di prova) e `pila.h` (che contiene la definizione delle funzioni da implementare) per testare il codice scritto. Infine, caricare **solo** il file `pila.cc` nello spazio apposito;
- Non modificare i file `esercizio3.cc` e `pila.h`. In altre parole, l'implementazione delle funzioni deve essere compatibile con il codice scritto in `esercizio3.cc` e `pila.h`, poichè questi due file verranno usati per valutare la correttezza dell'esercizio;
- All'interno del file `pila.cc` **non é ammesso** l'utilizzo di funzioni di libreria al di fuori di quelle definite in `iostream`;
- All'interno del file `pila.cc` **é ammesso** l'utilizzo di variabili di tipo `static`, che possono essere usate per mantenere informazioni sullo stato delle pile;
- Ricordarsi di deallocare la memoria allocata dinamicamente;
- Ricordarsi di distinguere gli esempi nella descrizione dell'esercizio (che servono solo ad aiutare a comprendere il problema) dalle istruzioni di implementazione;
- Per essere considerata unica, una lista doppiamente concatenata deve permettere, a partire da un **qualsiasi** nodo, di visitare ogni altro nodo della lista in maniera contigua.

Information for graders:

Total of marks: 40