Università di Catania
Dipartimento di Matematica e Informatica
Corso di Studio in Informatica, A.A. 2023-2024
Prova di Laboratorio
31 Gennaio 2024
Compito A

Descrizione del programma

Scrivere un programma in C che:

- **A.** Prenda un input da tastiera (argomenti della funzione main) costituito da due caratteri **a** e **b**, due numeri interi **x** e **y**, ed un numero intero **n**. Tali parametri devono rispettare le seguenti specifiche:
  - I due caratteri a e b devono appartenere al range [‘a’,’z’] oppure [‘A’, ‘Z’] (quindi entrambi minuscoli o entrambi maiuscoli); inoltre deve essere a<=b. Quindi ad esempio a=’C’, b=’Y’ oppure a=’d’, b=’h’;
  - numeri x e y devono appartenere al range [5,30], ed inoltre deve essere y-x>5;
  - il numero n deve appartenere al range [15,25].

- B. Produca n stringhe che rispettino le seguenti specifiche:
  - Ogni stringa sia generata con caratteri pseudo-casuali appartenenti al range [a,b];
  - la lunghezza di ogni stringa, sia L, dovra’ essere un numero subpseudo-casuale appartenente al range [x,y] (per ogni stringa sara’ generato un nuovo numero L in [x,y]).

- **C.** inserisca ogni stringa in una struttura dati dinamica **CODA**, e le stampi sullo standard output.

- D. Rimuova dalla coda tutti gli elementi e stampi sullo standard output le stringhe contenute all’interno di tali elementi, come specificato nel successivo punto E.

- E. Con riferimento al punto D, in fase di output, le vocali di ogni stringa andranno sostituite con il carattere ‘X’ (x maiuscolo).

Specifiche

Il programma potra’ essere strutturato in un unico file sorgente, ma dovra’ contenere almeno le seguenti funzioni:

1. **readInput()**: funzione che prende in input l'array di puntatori a carattere argv ed il numero di argomenti argc della funzione main, controlla che gli argomenti richiesti siano nel numero e nei limiti specificati, e restituisca parametri {x,y,a,b,n} in una struct da restituire al chiamante.

2. **genString()**: funzione con opportuni parametri formali, che produca una stringa sulla base delle specifiche descritte nel punto B. NB: **usare la funzione di generazione di numeri pseudo-casuali riportata in seguito nel testo (get_random()).**

3. **enqueue() e dequeue()**: funzioni per la gestione della coda;

4. **buildQueue()**: funzione che crea una coda di stringhe sulla base delle specifiche presenti al punto C, utilizzando opportunamente la funzione genString() e la funzione enqueue(). La funzione dovra' stampare tutte le stringhe generate e inserite nella cosa.
5. **printStrings()**: funzione che rimuove tutti gli elementi dalla coda mediante la funzione deQueue() e che stampi le stringhe nel modo specificato nel punto E mediante la funzione elabString() descritta al prossimo punto.

6. **elabString()**: funzione che prenda in input una stringa e modifichi la stringa (o restituisca una copia) in cui le vocali sono state sostituite con il carattere 'X' (x maiuscolo).

È VIETATO usare variabili globali.

Durata della prova: 120 minuti. NB: Inserire nome, cognome e numero di matricola all'interno del file sorgente.

Generazione di numeri pseudocasuali:

- Si consideri la seguente funzione get_random() per la generazione di numeri pseudo-casuali interi positivi (qualora necessaria):

  https://pastebin.com/f6eAKNQy

```
unsigned int get_random() {
	static unsigned int m_w = 123456;
	static unsigned int m_z = 789123;
	m_z = 36969 * (m_z & 65535) + (m_z >> 16);
	m_w = 18000 * (m_w & 65535) + (m_w >> 16);
	return (m_z << 16) + m_w;
}
```

NB: Ai fini della eventuale generazione di numeri in virgola mobile, si faccia uso della costante UINT_MAX (<limits.h>) unitamente alla funzione get_random().

Input e output di test (output.txt nella home directory della macchina virtuale):

(./a.out <a> <b> <x> <y> <n>)

```
$./a.out a p 10 20 23


buildQueue(), (i=0): eifohdelclnbe
buildQueue(), (i=1): aphcchfhlkfbmedb
buildQueue(), (i=2): epoldpggjighlhbp
buildQueue(), (i=3): edeobanankkamgfned
buildQueue(), (i=4): ignooepdeliihpameeaf
buildQueue(), (i=5): phcigoppgifjpf
buildQueue(), (i=6): ocekgdfiekhoahli
buildQueue(), (i=7): ncbjaihino
buildQueue(), (i=8): jecdncgpgengkcfkd
buildQueue(), (i=9): bbcdekkobnabnemhcmd
buildQueue(), (i=10): jjbpdebmaaiehcijbhlk
buildQueue(), (i=11): iehajkjmemkbghlplpk
buildQueue(), (i=12): lodpkiiamfdeb
buildQueue(), (i=13): olddmlmpcfafiedanjnd
buildQueue(), (i=14): nighahbhkbeko
buildQueue(), (i=15): ccemeipeagdieimnk
buildQueue(), (i=16): blmkmlfiiobpahnhimao
buildQueue(), (i=17): comdkchbieankcacffcj
buildQueue(), (i=18): bdginhokbmlkcgp
buildQueue(), (i=19): pmbhoacefdlbd
buildQueue(), (i=20): kfbmhpgoba
buildQueue(), (i=21): albhkhfdmocpgoggldma
```

```
buildQueue(), (i=22): fcmkgllaliocbjb

printStrings(), (i=0) XXfXhdXlclnbX
printStrings(), (i=1) XphcchfhlkfbmXdb
printStrings(), (i=2) XpXldpggjXghlhbp
printStrings(), (i=3) XdXXbXnXnkkXmgfnXd
printStrings(), (i=4) XgnXXXpdXlXXhpXmXXXf
printStrings(), (i=5) phcXgXppgXfjpf
printStrings(), (i=6) XcXkgdfXXkhXXhlX
printStrings(), (i=7) ncbjXXhXnX
printStrings(), (i=8) jXcdncgpgXngkcfkd
printStrings(), (i=9) bbcdXkkXbnXbnXmhcmd
printStrings(), (i=10) jjbpdXbmXXXXhcXjbhlk
printStrings(), (i=11) XXhXjkjmXmkbghlplpk
printStrings(), (i=12) lXdpkXXXmfdXb
printStrings(), (i=13) Xlddmlmpcf XfXXdXnjnd
printStrings(), (i=14) nXghXhbhkbXkX
printStrings(), (i=15) ccXmXXpXXgdXXXmnk
printStrings(), (i=16) blmkmlfXXXbpXhnhXmXX
printStrings(), (i=17) cXmdkchbXXXnkcXcffcj
printStrings(), (i=18) bdgXnhXkbmlkcgp
printStrings(), (i=19) pmbhXXcXfdlbd
printStrings(), (i=20) kfbmhpgXbX
printStrings(), (i=21) XlbhkhfdmXcpgXggldmX
printStrings(), (i=22) fcmkgllXlXXcbjb
```