[1]

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

https://pastebin.com/f6eAKNQy

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

Si scriva un programma C come segue:

- **A (8 punti)**. Il programma prenda un input da tastiera (argomenti della funzione main()) un intero k in [10,15], un carattere w in [‘a’-’z’] e due interi N ed M in [4,8]; se gli argomenti a riga di comando non rispondono ai suddetti requisiti, il programma stampa un messaggio di errore sullo standard error e termina la propria esecuzione;
- **B (4 punti).** Allocazione dinamica di una matrice (sia S) di stringhe (char *) di dimensioni N x M;
- **C (5 punti).** Si riempia la matrice S con NxM stringhe di lunghezza k composte di caratteri pseudo-casuali in [a-z];
- **D (6 punti).** Si ordini ogni colonna della matrice S in modo crescente (ordinamento lessicografico) con un algoritmo di ordinamento a scelta tra Insertion Sort e Bubble Sort.
- **E (2 punti)**. Si stampi la matrice sullo standard output.
- **F (5 punti).** Si stampi sullo standard output la stringa (e gli indici all'interno della matrice) che contiene il maggior numero di occorrenze del simbolo w. Queste ultime vanno sostituite, sullo standard output, con il carattere '*'.

Il programma potra' essere articolato in un unico file sorgente, ma dovra' contenere <u>almeno le seguenti funzioni con opportuni parametri formali:</u>

- **puntoA()**: funzione che prenda il vettore argv ed il numero argc della funzione main(), che controlli la presenza ed i requisiti degli argomenti k, w, N ed M, e che li inserisca in un record (struct) da restituire allo user code (funzione main);
- **puntoB()**: - funzione per allocazione dinamica della matrice di dimensioni NxM, tale matrice va restituita come dato di ritorno al chiamante (funzione main()).
- **genString()** - restituisca una stringa della lunghezza specificata con caratteri pseudo-casuali in un ben determinato insieme specificato mediante opportuni parametri formali;
- **puntoC()** - funzione di riempimento della matrice S come specificato nel punto C;
- **puntoD() -** funzione di ordinamento della matrice come specificato nel punto D; NB: si faccia uso, al suo interno, della funzione di libreria strcmp();
- **puntoE()** - funzione per il punto E;
- **puntoF()** - funzione per il punto F.

VIETATO usare variabili globali.

[2]

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

https://pastebin.com/f6eAKNQy

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

Si scriva un programma C come segue:

- **A (8 punti).** Il programma prenda un input da tastiera (argomenti della funzione main()) un intero positivo N in [10,20] ed un carattere w; se gli argomenti a riga di comando non rispondono ai suddetti requisiti, il programma stampa un messaggio di errore sullo standard error e termina la propria esecuzione;
- **B (16 punti)** si generi, mediante successivi inserimenti, una lista concatenata semplice (ordinata in modo crescente - ordinamento lessicografico) che contenga N stringhe con caratteri pseudo-casuali in [‘a’-’z’] di lunghezza pseudo-casuale L nell'intervallo [5,15];
- **C (3 punti)** il programma stampi sullo standard output l'intera lista;
- **D (3 punti)** il programma stampi sullo standard output il numero totale di occorrenze del carattere w in tutte le stringhe della lista.

Il programma potra' essere articolato in un unico file sorgente, ma dovra' contenere almeno le seguenti funzioni (con opportuni parametri formali):

- **puntoA() -** funzione che prenda in input gli argomenti della funzione main argv ed argc, controlli i requisiti dei parametri e restituisca al chiamante un record (struct) che contenga tutti i parametri.
- **genString()**: funzione che restituisca una stringa di caratteri pseudo-casuali appartenenti ad un determinato intervallo specificato mediante opportuni parametri formali.
- **insertString():** funzione che inserisca una stringa nella lista; per tale funzione e’ possibile usare la funzione di libreria strcmp();
- **puntoB():** funzione che si occupa di creare e riempire la lista di stringhe come richiesto nel punto B.
- **puntoC().**
- **puntoD().**

VIETATO usare variabili globali.

[3]

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

https://pastebin.com/f6eAKNQy

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

Si scriva un programma C come segue:

- **A (8 punti).** Il programma prenda un input da tastiera (argomenti della funzione main) costituito da un intero positivo N in [10,20], e due numeri in virgola mobile positivi x,y. Dovra' essere 5.0 < x < y < 30.0; se gli argomenti a riga di comando non rispondono ai suddetti requisiti, il programma stampa un messaggio di errore sullo standard error e termina la propria esecuzione.
- **B (3 punti).** Allocazione dinamica di una matrice (sia A) di numeri double pseudo-casuali in [x,y] di dimensioni N x N.
- **C (5 punti)**. Si calcoli il minimo degli elementi della diagonale principale della matrice (sia minp), ed il massimo valore degli elementi della diagonale secondaria della matrice stessa (sia maxd); si restituisca inoltre il numero di elementi della matrice aventi valori in [mind, maxd].
- **D (4 punti).** Allocazione dinamica di un array di double e riempimento con tutti gli elementi di A nell'intervallo [minp, maxd].
- **E (5 punti)**. Ordinamento dell'array mediante un algoritmo a scelta tra selection sort e insertion sort.
- **F (2 punti).** Si stampi l'array sullo standard output, insieme allla media aritmetica dei suoi elementi.

Il programma potra' essere articolato in un unico file sorgente, ma dovra' contenere almeno le seguenti funzioni (con opportuni parametri formali):

- **puntoA()**: funzione che prenda in input gli argomenti argv ed argc della funzione main(), controlli i requisiti dei parametri e restituisca un record che contiene tali parametri, altrimenti (se i requisiti non sono soddisfatti), NULL.
- **genDouble():** genera un numero double in un intervallo specificato mediante parametri formali;
- **puntoB()**: funzione che crea e restituisce una matrice con elementi double generati dalla funzione genDouble().
- **puntoC():** calcola mind e maxD e restituisce il numeri di elementi che ricadono nell'intervallo [mind, maxd] come specificato nel punto C.
- **puntoD():** creazione e riempimento array come specificato nel punto D.
- **puntoE():** ordinamento dell'array come specificato nel punto E.
- **puntoF()**: stampa dell'array come specificato nel punto F.

VIETATO usare variabili globali.

[4]

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

https://pastebin.com/f6eAKNQy

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

Si scriva un programma C come segue:

- **A (8 punti) -** il programma prenda un input da tastiera (argomenti della funzione main) costituito da intero positivo N in [20,25], un intero positivo M in [10,15] e due numeri in virgola mobile positivi x,y tali che x in [5.0, 10.0], y in [40, 50]. Se gli argomenti a riga di comando non rispondono ai suddetti requisiti, il programma stampa un messaggio di errore sullo standard error e termina la propria esecuzione.
- **B (15 punti) -** il programma produca una sequenza di operazioni di inserimento (push()) di numeri double casuali in [x,y] ed operazioni pop() in una struttura dati LIFO (stack o pila) di dimensione massima N/2 da implementare mediante array, nel seguente modo:
    - lo stack (sia S1) sara' rappresentato da una struct che contiene almeno le seguenti informazioni: array di double + informazione che rappresenta il top dello stack (indice o puntatore);
    - la prima operazione sara' di inserimento (push());
    - ogni qual volta il numero inserito (sia w) e' tale che w >= (x+y)/2, allora la successiva operazione sara' una rimozione dallo stack (pop());
    - l'operazione successiva ad ogni rimozione (pop()) sara' un inserimento (push());
    - andranno eseguite esattamente N operazioni di inserimento, mentre le rimozioni pop() potranno essere al piu' M;
    - se si raggiunge il numero massimo di rimozioni (pop()), le successive operazioni saranno inserimenti fino al numero massimo N;
    - per ogni operazione di pop(), l'elemento rimosso andra' stampato sullo standard error;
    - un tentativo di inserimento in uno stack pieno termina la procedura;
- **C (5 punti)** si modifichi la procedura al punto B in modo da inserire ogni elemento rimosso dallo stack del punto precedente in un altro analogo stack (sia S2) di dimensione massima N/2;
- **D (2 punti)** si stampi sullo standard output il contenuto di S1 ed S2, dal top alla base;

Il programma potra' essere articolato in un unico file sorgente, ma dovra' contenere almeno le seguenti funzioni (con opportuni parametri formali):

- **puntoA():** una funzione che prenda in input i parametri argv e argc della funzione main e che restituisca un record (struct) con i parametri o, in caso i requisiti dei parametri non siano soddisfatti un puntatore NULL;

- **buildStack()**: alloca, inizializza e restituisce uno stack di double di dimensione massima specificata mediante parametri formali;
- **push() e pop()**: funzioni che consentono di inserire un elemento sullo stack di double o rimuovere un elemento da esso.
- **genDouble():** funzione che produce un numero double pseudo-casuale in un certo intervallo specificato mediante parametri formali.
- **puntoB():** funzione che rappresenta l'implementazione del punto B.
- **puntoD()**: per la stampa del contenuto di S1 ed S2.

VIETATO usare variabili globali.

[5]

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

https://pastebin.com/f6eAKNQy

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

Si scriva un programma C come segue:

- **A (8 punti)** il programma prenda un input da tastiera (argomenti della funzione main()) costituito da un intero positivo N in [5,9] e due caratteri minuscoli, siano v e w; v dovra' essere una vocale, mentre w dovra' essere una consonante; se gli argomenti a riga di comando non rispondono ai suddetti requisiti, il programma stampa un messaggio di errore sullo standard error e termina la propria esecuzione;
- **B (15 punti)** il programma generi una sequenza di operazioni di inserimento (push()) di caratteri pseudo-casuali in [A-Z,a-z,1-9] in una struttura dati LIFO dinamica (pila o stack, sia S1) da implementare mediante lista concatenata semplice (top==testa della lista), nel seguente modo:
    - ad ogni passo, si generi innanzitutto un carattere x in [1-9];
    - se x rappresenta un numero in [1-4], allora si proceda ad x operazioni di inserimento (push()) di caratteri pseudo-casuali che siano vocali, seguite dall'inserimento del carattere x;
    - se x rappresenta un numero in [5-9], allora si proceda ad x operazioni di inserimento (push()) di caratteri pseudo-casuali che siano consonanti, seguite dall'inserimento del carattere x;
    - in particolare, sia c il generico carattere da inserire sullo stack:
        - se c==v, si inserisca sullo stack al posto di esso il carattere '*';
        - se c==w, si inserisca sullo stack al posto di esso il carattere '?';
    - infine, si ponga sul top dello stack il numero N.

- **C (5 punti)**
    - si crei un array di stringhe (puntatori a caratteri) di lunghezza N;
    - poi si proceda nel seguente modo, fino a svuotamento dello stack: successivamente si proceda ad una sequenza di operazioni di rimozione (pop()) come segue:
        - la prima operazione di rimozione trovera' sullo stack il carattere corrispondente al numero N;
        - si proceda quindi, ad ogni passo con un'operazione di rimozione (pop()) del carattere x che indica (per costruzione) il numero di caratteri da rimuovere successivamente mediante x operazioni pop();
        - si memorizzi sull'array di caratteri ogni stringa composta dal carattere x e dai successivi x caratteri;

- **D (2 punti)** si stampi, sullo standard output, il contenuto dell'array di stringhe;

Il programma potra' essere articolato in un unico file sorgente, ma dovra' contenere almeno le seguenti funzioni (con opportuni parametri formali):

- **puntoA():** funzione che prende in input l'array di puntatori a carattere argv della funzione main, controlli che gli argomenti richiesti siano nei limiti specificati, e restituisca il record (struct) che contiene tali parametri; se il controllo non va a buon fine, stampa un messaggio sullo standard error e restituisce NULL.
- **genVowel():** funzione che produca un carattere vocale pseudo-casuale;
- **genConsonant():** funzione che produca un carattere consonante pseudo-casuale;
- **push() e pop():** funzioni che consentono di inserire un elemento sullo stack o rimuovere un elemento da esso;
- **puntoB():** funzione con opportuni parametri formali che rappresenti l'implementazione della procedura descritta nel punto B;
- **puntoC():** funzione con opportuni parametri formali che sia conforme alla procedura descritta nel punto C;
- **puntoD():** funzione per la stampa dell'array di stringhe prodotto al punto C.

VIETATO usare variabili globali.

---

[5]

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

https://pastebin.com/f6eAKNQy

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

Si scriva un programma C come segue:

- **A (8 punti)** il programma prenda un input da tastiera (argomenti della funzione main()) costituito da un intero positivo
- **(15 punti)** il programma generi una sequenza di operazioni di inserimento (push()) di caratteri pseudo-casuali in [A-Z,a-z,1-9] in una struttura dati LIFO dinamica (pila o stack, sia S1) da implementare mediante lista concatenata semplice (top==testa della lista), nel seguente modo:
    - ad ogni passo, si generi innanzitutto un carattere x in [1-9];
    - se x rappresenta un numero in [1-4], allora si proceda ad x operazioni di inserimento (push()) di caratteri pseudo-casuali che siano vocali, seguite dall'inserimento del carattere x;
    - se x rappresenta un numero in [5-9], allora si proceda ad x operazioni di inserimento (push()) di caratteri pseudo-casuali che siano consonanti, seguite dall'inserimento del carattere x;
    - in particolare, sia c il generico carattere da inserire sullo stack:
        - se c==v, si inserisca sullo stack al posto di esso il carattere '*';
        - se c==w, si inserisca sullo stack al posto di esso il carattere '?';
    - infine, si ponga sul top dello stack il numero N.

- **C (5 punti)**
    - si crei un array di stringhe (puntatori a caratteri) di lunghezza N;
    - poi si proceda nel seguente modo, fino a svuotamento dello stack: successivamente si proceda ad una sequenza di operazioni di rimozione (pop()) come segue:
        - la prima operazione di rimozione trovera' sullo stack il carattere corrispondente al numero N;
        - si proceda quindi, ad ogni passo con un'operazione di rimozione (pop()) del carattere x che indica (per costruzione) il numero di caratteri da rimuovere successivamente mediante x operazioni pop();
        - si memorizzi sull'array di caratteri ogni stringa composta dal carattere x e dai successivi x caratteri;

- **D (2 punti)** si stampi, sullo standard output, il contenuto dell'array di stringhe;

Il programma potra' essere articolato in un unico file sorgente, ma dovra' contenere almeno le seguenti funzioni (con opportuni parametri formali):

- **puntoA():** funzione che prende in input l'array di puntatori a carattere argv della funzione main, controlli che gli argomenti richiesti siano nei limiti specificati, e restituisca il record (struct) che contiene tali parametri; se il controllo non va a buon fine, stampa un messaggio sullo standard error e restituisce NULL.
- **genVowel():** funzione che produca un carattere vocale pseudo-casuale;
- **genConsonant():** funzione che produca un carattere consonante pseudo-casuale;
- **push() e pop():** funzioni che consentono di inserire un elemento sullo stack o rimuovere un elemento da esso;
- **puntoB():** funzione con opportuni parametri formali che rappresenti l'implementazione della procedura descritta nel punto B;
- **puntoC():** funzione con opportuni parametri formali che sia conforme alla procedura descritta nel punto C;
- **puntoD():** funzione per la stampa dell'array di stringhe prodotto al punto C.

VIETATO usare variabili globali.
