Dipartimento di Elettronica e Informazione
Politecnico di Milano
20133 Milano (Italia)
Piazza Leonardo da Vinci, 32
Tel. (39) 02-2399.3400
Fax (39) 02-2399.3411

Prof. Luca Mottola

Fondamenti di Informatica (ELT)

8 Luglio 2015

Cognome
Nome
Matricola

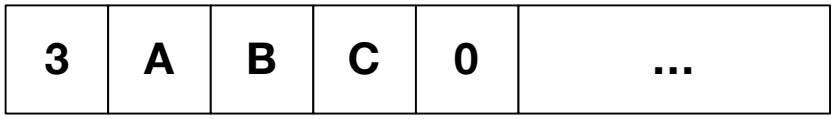
Note

1. La mancanza dei dati di cui sopra comporta l'annullamento della prova.
2. Restituire **solo** queste pagine. Verranno ignorati fogli aggiuntivi. **Non** si può scrivere con la matita.
3. È proibito l'uso di dispositivi elettronici (computers, calcolatrici, cellulari, macchine fotografiche,…).
4. Qualunque frammento di codice richiesto deve essere scritto in **stampatello**.
5. Sebbene non obbligatorio, si consiglia di inserire commenti nel codice per facilitarne la correzione.
6. Non si può tenere una copia del testo dell'esame quando si lascia l'aula.
7. Un punteggio inferiore a 17 preclude la registrazione del voto complessivo anche se sufficiente.
8. Il tempo a disposizione è di 2 ore.

Domanda 1: Macchina di Von Neumann (8 punti)

Si descriva il funzionamento dell'algoritmo che viene implementato dal seguente programma per la macchina di Von Neumann. Si descriva inoltre il ruolo delle celle 101, 102, 103, e 104 nel funzionamento di tale algoritmo. Per risolvere l'esercizio, si consiglia (ma non è obbligatorio) di studiare il funzionamento del programma quando il nastro di ingresso contiene i simboli in figura.

1. READ
2. STORE 101
3. LOAD= 110
4. STORE 102
5. READ
6. BEQ 12
7. STORE@ 102
8. LOAD 102
9. ADD= 1
10. STORE 102
11. BR 5
12. LOAD= 0
13. STORE 103
14. LOAD 101
15. SUB 103
16. BEQ 32
17. LOAD= 110
18. STORE 104
19. LOAD 102
20. SUB 104
21. BEQ 28
22. LOAD@ 104
23. WRITE
24. LOAD 104
25. ADD= 1
26. STORE 104
27. BR 19
28. LOAD 103
29. ADD= 1
30. STORE 103
31. BR 14
32. END



Domanda 2: Linguaggio C – prima parte (6 punti)

Si consideri un tipo di file dove è memorizzata una stringa di caratteri. Si scriva una procedura in linguaggio C che prenda come parametro il nome del file e legga dal file la stringa carattere per carattere. Poiché la lunghezza della stringa non è nota a priori, la procedura deve memorizzare i caratteri letti dal file in una lista dinamica di caratteri.

Per la lettura dal file, si utilizzino le seguenti funzioni messe a disposizione dalle librerie C:

- `FILE *fopen(char *filename, char *mode);`
- `void fclose(FILE *stream);`
- `char fgetc(FILE *stream);`

Per la memorizzazione nella lista dinamica, si assuma di poter usufruire (senza implementarle) delle seguenti funzioni:

- `lista_char* creaNuova();//`per creare una nuova lista
- `lista_char* aggiungiCoda(lista_char* lista, char c); //`per aggiungere un nuovo elemento in coda alla lista

Nel caso si ritenga che altre funzioni siano necessarie, definirne dapprima il prototipo e giustificarne la necessità. Non è richiesta l'implementazione delle funzioni aggiuntive.

Domanda 3: Linguaggio C – seconda parte (8 punti)

Si definisca esplicitamente il tipo `lista_char` della domanda 3. Implementare inoltre le funzioni `creaNuova` e `aggiungiCoda` dell'esercizio 3 e in aggiunta le seguenti funzioni:

- `int conta(lista_char* lista); //`che ritorna il numero di elementi nella lista
- `char dammiElemento(lista_char* lista, int i); //`che ritorna l'elemento i-esimo

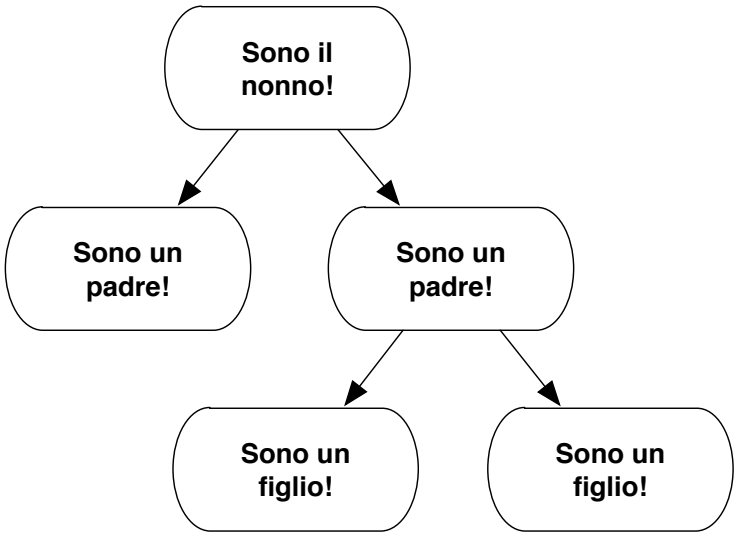
Utilizzando le funzioni di cui sopra come si preferisce, implementare un'ulteriore funzione

```
int palindroma(lista_char* lista);
```

che verifica se la stringa di caratteri memorizzata nella lista è palindroma o meno.

Domanda 4: Processi (8 punti)

La figura a lato mostra una possibile gerarchia di processi, ciascuno con un messaggio che viene stampato a video dal processo stesso.



Si implementi un frammento di codice in linguaggio C che crei esattamente questa gerarchia di processi e faccia sì che tutti i processi ad uno stesso livello della gerarchia stampino i propri messaggi prima che uno qualsiasi dei processi al livello immediatamente superiore stampi il proprio messaggio.

In altre parole, l'output che ci si aspetta è ad esempio:

```
Sono un figlio!
Sono un figlio!
Sono un padre!
Sono un padre!
Sono il nonno!
```

Mentre, ad esempio, un output come il seguente non è considerato corretto, poiché un padre ha stampato prima di uno dei figli.

```
Sono un figlio!
Sono un padre!
Sono un figlio!
Sono un padre!
Sono il nonno!
```

Si ricordano di seguito i prototipi delle funzioni *eventualmente* rilevanti per l'esercizio:

- `pid_t fork();`
- `void exit (int stato);`
- `pid_t getpid();`
- `pid_t wait(int* stato);`
- `pid_t waitpid(pid_t pid, int* stato);`