Domino Lineare

Il Domino Lineare si gioca usando di 21 tessere, ciascuna con una coppia di numeri da 1 a 6. Se raffiguriamo una tessera tra parentesi quadre, sotto tutte le tessere del domino.

```
[1|1] [1|2] [1|3] [1|4] [1|5] [1|6]
[2|2] [2|3] [2|4] [2|5] [2|6]
[3|3] [3|4] [3|5] [3|6]
[4|4] [4|5] [4|6] [5|5] [5|6] [6|6]
```

Al giocatore vengono assegnate `N` di queste tessere, anche con *ripetizioni*, ed il suo obiettivo è di disporle sul piano di gioco inizialmente vuoto totalizzando il massimo punteggio. Ad esempio il giocatore potrebbe ricevere le seguenti `N=10` tessere:

```
[6|6] [6|6] [6|6] [3|6] [1|6] [2|6] [1|4] [3|4] [4|4] [4|5]
```

Le tessere possono essere posizionate solo in linea orizzontale. La prima tessera è scelta liberamente dal giocatore. Ad esempio:

```
[6|6]
```

Il giocatore può adesso porre un'altra delle sue tessere sul piano di gioco, sempre in linea orizzontale, a patto che le cifre di due tessere adiacenti siano uguali. Le tessere possono essere poste sia sul lato destro che sul lato sinistro. Ad esempio, giocatore può aggiungere la tessera `[2|6]` a sinistra e un'altra tessera `[6|6]` a destra come sotto.

```
[2|6][6|6][6|6]
```

Il giocatore può ruotare le tessere in suo possesso. Ad esempio aggiungendo la tessera `[1|6]` a destra come sotto.

```
[2|6][6|6][6|6][6|1]
```

Quando il giocatore non può più posizionare tessere il gioco termina. Il punteggio del giocatore è dato dalla somma delle cifre nelle tessere poste sul piano di gioco. Se il gioco terminasse qui, il punteggio sarebbe quindi di $2 + 6 + 6 + 6 + 6 + 6 + 1 + 6 = 39$ punti.

Modalità interattiva

Viene chiesto all'utente quale sia la mossa successiva. Il piano di gioco ed il punteggio corrente vengono aggiornati di conseguenza. Il gioco termina automaticamente se il giocatore non ha più mosse valide disponibili.

Modalità AI

Si richiede di implementare una funzione che calcoli una buona strategia di gioco.

Strategia 1: First Match

Provo tutte le tessere in mio possesso e posiziono la prima che abbia un match.

Altre Strategie

A voi piena libertà. Ad esempio preferire le tessere che danno un punteggio più alto, o con cifre che appaiono frequentemente in altre tessere in possesso del giocatore. Un'opzione è una soluzione ricorsiva che implementi la ricerca esaustiva della migliore configurazione possibile.

Partite Speciali

Potete pensare a partite che iniziano da un insieme specifico di pezzi per il giocatore (e non casuale). Durante il corso verranno pubblicati alcuni di questi insiemi e vi verrà chiesto di trovare la soluzione migliore.

Schema di implementazione

Il programma che implementerete dovrà chiedere al giocatore quale mossa giocare, e dovrà visualizzare il campo di gioco aggiornato (usando la funzione `printf` o equivalente). Dovrà verificare che la mossa scelta sia legale (ci sia un match tra le cifre), e dovrà aggiornare il punteggio. Questi passi si ripetono fino alla fine della partita.

Nella modalità AI, il programma dovrà calcolare le mosse migliori, e visualizzarle a schermo assieme al piano di gioco finale.

Siete liberi di sperimentare e inventare schemi di gioco o varianti a vostro piacimento, pur rispettando i requisiti riportati più in basso in questo documento.

Consegna

Quando? Il progetto deve essere consegnato 4 giorni prima la data di ciascun appello. E' sempre obbligatorio iscriversi all'appello di "Esercizi" del prof. Spanò per sostenere la discussione del progetto.

Dove? Il docente abiliterà la consegna tramite Moodle secondo le scadenze previste.

Cosa? Dovrete consegnare un unico file zip contenente:

1. Una relazione scritta di al massimo 3 pagine che descriva la struttura del vostro progetto, l'organizzazione del lavoro tra i componenti del gruppo, le principali difficoltà incontrate. Relazioni più lunghe verranno penalizzate.
2. Il codice sorgente del progetto in linguaggio C99; eventuali parti di codice scritto in altri dialetti vanno isolati in sorgenti separati ed il progetto dovrà compilare opportunamente.
3. Documentazione delle funzioni, dei tipi e dei file generata con Doxygen

Librerie esterne

È possibile implementare l'intero gioco utilizzando semplicemente la standard library di C, usando le funzioni di lettura dei tasti come `scanf`, `getc` o `getchar` e le funzioni di stampa come `printf`. Coloro che vogliono cimentarsi nell'uso di librerie esterne per dare una veste grafica ed una interazione col giocatore più accattivanti e fluide possono farlo, tuttavia sarà propria responsabilità occuparsi degli aspetti di portabilità e compatibilità. Il programma deve compilare e funzionare correttamente sulle 3 piattaforme (Windows, Linux, Mac); in caso contrario, è necessario documentare e motivare opportunamente la scelta nella relazione che accompagnerà il progetto ed in sede d'esame.

Valutazione

Per i progetti di *gruppo*:

- progetto **sufficiente** se permette ad un utente di giocare inserendo le proprie mosse tramite input da tastiera (modalità interattiva); l'implementazione delle regole del gioco deve essere corretta.
- progetto **buono** se implementa la modalità AI utilizzando un algoritmo semplice; avete piena libertà di definire una strategia.

- progetto **ottimo** se implementa una strategia particolarmente interessante, come ad esempio un **algoritmo ricorsivo**, per il la modalità AI;

Tutti i membri del gruppo devono conoscere ogni riga del codice!

Nel caso di progetti *individuali*, ad esempio per studenti lavoratori:

- progetto **sufficiente** se permette ad un utenti di giocare inserendo le proprie mosse tramite input da tastiera; l'implementazione delle regole del gioco può essere ragionevolmente parziale.
- progetto **buono** se permette ad un utente di giocare inserendo le proprie mosse tramite input da tastiera; l'implementazione delle regole deve essere corretta.
- progetto **ottimo** se implementa una strategia in modalità AI; avete piena libertà di definire una strategia.

Importante! Il progetto è di gruppo, ma **la valutazione è individuale**. Questo significa che i componenti di un gruppo potrebbero ricevere un voto diverso.

Inoltre, è possibile per alcuni dei componenti presentare individualmente delle **migliorie** al progetto. Possibili migliorie potrebbero essere: grafica migliore, strategia di gioco più raffinata, menu di interazione dell'utente più usabile, ecc.

Non ci sono limiti alle aggiunte o modifiche che vorrete fare! Quindi non ponetevi limiti!

Challenges

Durante il corso vi proporremo alcune configurazioni di pezzi su cui sperimentare la vostra strategia AI. **Aggiungeremo inoltre pezzi speciali e varianti del gioco.**

Fisseremo ~~3~~ 2 scadenze intermedie:

- 17 Novembre
- ~~01 Dicembre~~
- 15 Dicembre

Il vostro codice verrà testato in modo da calcolare il punteggio della vostra soluzione.

Sarà un ottimo modo per avere dei commenti sul progresso del vostro lavoro e il vostro risultato verrà preso in considerazione nella discussione del progetto.

Specifiche per le challenges

Per le challenges il codice verrà testato in maniera *automatica*, pertanto la vostra implementazione dovrà essere molto precisa e rispettare il protocollo seguente.

Consegna. Dovrete consegnare in Moodle un unico file `nome-gruppo.zip`, contenente un'unica cartella denominata `iap` con al suo interno tutti i file sorgente. Basta che un solo membro del gruppo effettui la consegna in Moodle. Nella cartelal `iap` deve essere presente un file `main.c` e la prima riga di questo file deve indicare i componenti del gruppo (anche nel caso di consegna individuale) come segue:

```
// matricola1 matricola2 matricola3
```

Compilazione. Il vostro codice verrà compilato con il comando `gcc -O2 -std=c99 --pedantic *.c -o iap`. Questo significa che potrete usare anche più di un unico file `.c`, e che non è concesso usare altre estensioni diverse da `.c`. Il comando sopra genera i file eseguibile `iap`.

Esecuzione. Il vostro eseguibile verrà invocato con un unico parametro addizionale a linea di comando come segue: `./iap --challenge`. Il suggerimento è di implementare un unico progetto, dove il parametro

`--challenge` abilita la consegna per le challenge, mentre l'esecuzione interattiva/AI avviene in assenza del parametro.

Una volta eseguito, verrà valutata la correttezza della vostra soluzione e verrà calcolato il punteggio corrispondente.

INPUT: Tessere assegnate. Nella modalità challenge, il vostro codice dovrà eseguire diverse `scanf` per leggere la sequenza di tessere in input. La prima `scanf` legge un intero N corrispondente al numero di tessere a disposizione dell'utente. Successivamente dovranno essere eseguite $2 \times N$ `scanf` per leggere le N coppie di numeri corrispondenti alle tessere. Un esempio di input è dato di seguito. *Suggerimento:* per testare velocemente il vostro codice vi consigliamo di "copia-incollare" le 11 righe di seguito nel terminale non appena viene richiesto il primo input.

```
10
6 6
6 6
6 6
3 6
1 6
2 6
1 4
3 4
4 4
4 5
```

OUTPUT: Mosse del giocatore AI. L'unico output del vostro programma deve essere una singola linea con le mosse della vostra partita. Il formato da usare è una sequenza di triple formate da un carattere, uno spazio, un numero, uno spazio, un altro numero e uno spazio. Il primo carattere può essere `S` per *Start*, `L` per *Left* o `R` per *Right*. I numeri sono invece i numeri delle tessere così come devono essere posizionate. Usate solo lettere maiuscole e non aggiungere altro testo (es. `"La soluzione è:"`).

Le mosse discusse nell'esempio iniziale sono codificate così:

```
S 6 6 L 2 6 R 6 6 R 6 1
```

Specifiche per la prima challenge (17 Novembre)

Dovranno essere implementate le regole discusse sopra. Il vostro codice verrà valutato solo con un numero fisso di pezzi pari a 4. La priorità è quella di produrre in output una strategia corretta. Sotto un esempio di input e di output.

Input:

```
4
3 4
1 3
4 5
2 2
```

Output:

```
S 1 3 R 3 4 R 4 5
```

Il punteggio finale è di `1+3+3+4+4+5 = 20`.

Specifiche per la seconda challenge (15 Novembre)

In questa challenge vengono aggiunte le tessere "speciali" elencate di seguito:

- `[0|0]`: può essere accostata a qualunque altra tessera. *Esempio:* `[1|2][0|0][5|6]`
- `[11|11]`: somma 1 a tutte le cifre di tutte le tessere sul piano di gioco tranne il 6 che diventa 1. La tessera può essere posizionata in qualunque posizione e le sue cifre vegono sostituite con la cifra adiacente dopo averla incrementata di 1. *Esempio:* `[1|6][6|3][11|11]` diventa `[2|1][1|4][4|4]`
- `[12|21]`: copia "a specchio" la tessera adiacente. La tessera può essere posizionata in qualunque posizione e le sue cifre vegono sostituite con le cifre della tessera adiacente in ordine inverso. *Esempio:* `[1|2][2|3][12|21]` diventa `[1|2][2|3][3|2]`

Specifiche per il progetto finale (da consegnare all'appello con il prof. Spanò)

Il progetto finale non deve prevedere la modalità `--challenge` ma solo le modalità interattiva ed AI. Deve però implementare tutte le aggiunte identificate nelle challenge. Quindi, il giocatore riceverà ad inizio partita un insieme casuale di tessere che includono le tessere speciali della seconda challenge.

In aggiunta deve implementare la seguente funzionalità: **2D Domino**.

Oltre a posizionare le tessere in maniera orizzontale, è possibile posizionare le tessere in maniera verticale, ma solo agli estremi del piano di gioco attuale, e solo facendo "crescere" il domino verso il basso.

Ad esempio, partendo da:

```
[5|1][1|6][6|2]
```

potremmo aggiungere la tessera `[2|3]` a destra in verticale (invece che orizzontale) e ottenere:

```
[5|1][1|6][6|2][2:
               :3]
```

a questo punto abbiamo due opzioni a destra (invece di una) per far crescere il gioco in maniera orizzontale, ad esempio aggiungendo la tessera `[3|5]`

```
[5|1][1|6][6|2][2:
               :3][3|5]
```

dopo ancora la tessera `[2|2]` (l'ordine di questi due ultimi inserimenti è irrilevante)

```
[5|1][1|6][6|2][2:[2|2]
               :3][3|5]
```

lo schema può avere altri livelli, ad esempio aggiungendo le tessere `[5|1]` e `[1|6]`

```
[5|1][1|6][6|2][2:[2|2]
               :3][3|5][5:
                        :1][1:
                           :6]
```

Casi particolari

Uno stesso livello può avere più tessere verticali:

```
[5|1][1|6][6|2][2:[2|2][2|4][4|1][1:
               :3][3|5][5:        5][5|2][2:
                        :1][1:             6]
```

```
                          :6]
```

Lo sviluppo del domino può avvenire sia verso destra che verso sinistra:

```
[5|1][1|6][6|2][2:[2|2][2|4][4|1][1:
          [1|3]:3][3|5][5:         5][5|2][2:
                       :1][1:        [1|6]:6][6|2]
                          :6]
```

Immaginate che ciascuna tessera occupi due caselle di una scacchiera, e imponiamo il vincolo che due tessere non possono sovrapporsi. In questo senso, una raffigurazione più corretta sarebbe la seguente dove ogni tessera occupa sempre 4 caratteri:

```
[51][16][62]{2[22][24][41]{1
        [13]3}[35]{5      5}[52]{2
                  1}{1      [16]6}[62]
                    6}
```

Le tessere sono identificate, per semplicità, con dei numeri e raffigurate come `[xx|xx]`, ma siete liberi di visualizzarle come meglio credete. Ad esempio `[2|3]` e `[11|11]` potrebbero essere visualizzate con `{2 3}` e `{ +1 }` o come sotto:

```
 ------ -----
| 2  3 | + 1 |
 ------ -----
```

Nota: Nel caso ci fossere dei dettagli del gioco che possono essere interpretati in più modi, l'importante è che voi fissiate e chiariate le vostre scelte.