ESERCIZIO 1 Considerare un tabellone da gioco stile labirinto che possa essere descritto come una matrice di caselle

1.A (1 PUNTO) Realizzare la struct casella. Ogni casella è descritta dalla posizione (riga, colonna), e dallo stato (per es ”libero”, “muro”, “occupato”)

1.B (2 PUNTI) Realizzare la funzione booleana
bool Adiacente (casella c1, casella c2)
che restituisce true se c1 e c2 sono vicini (sinistra-destra o sopra-sotto)

ESERCIZIO 2 considerare un percorso sul labirinto, inteso come sequenza di caselle (utilizzare i vector)

2.A (2.5 PUNTI) produrre una funzione di inserimento di una casella nel percorso, dopo aver verificato che sia interna al tabellone e che sia adiacente alla casella precedente

2.B (2.5 PUNTI)
Realizzare una funzione che calcoli l’ampiezza massima del percorso (ossia la distanza dalla casella più a sinistra a quella più a destra)

ESERCIZIO 3 Considerare un’implementazione del tipo di dato insieme basata su liste semplici (tipo base: interi)

3.A (1 PUNTI) Definire il tipo di dato insieme

3.A (2.5 PUNTI) Realizzare la funzione che effettui l’inserimento di un elemento nuovo, nel rispetto delle proprietà degli insiemi

3.B (1.5 PUNTI) Realizzare una funzione ricorsiva che stampi gli elementi dell’insieme

Es 2.A Una soluzione possibile

- ```
  Void InserisciCasella (Vector<casella>&v, casella c) {
    if !(Interna(c)) throw OUT_OF_BOUND;
        if (Adiacente(v.at(v.size()),c)
      v.push_back(c);
  ```

- Adiacente e' la funzione realizzata nell'esercizio 1.B
- La funzione Interna verifica che la posizione di casella abbia righe e colonne maggiori o uguali a 0 e minori della dimensione della tabella (che potevate assumere definita da una costante globale const int DIM=...)
  ```

Es 2.B - e' una variante della ricerca di min e max in un vector

```
Int Ampiezza(vector<casella> v) {
     int left=DIM;
     int right=0;
     if v.empty() return 0;
     for (int i=0;i<v.size();++i)
         if (v.at(i)<left)
         left=v.at(i);
         if (v.at(i)>right)
         right=v.at(i);
             return right-left+1;
}
```

Es3

- E’ un classico esercizio sulle liste
- L’unica cosa da ricorcare e’ che gli insiemi non ammettono ripetizione di elementi