```
Esercizio 1 palude ricorsiva con prove di correttezza
#include<iostream>

using namespace std;

void leggi(bool *B, int nelem) {
    if(nelem>0)
       {
          cin >> B[0];
          leggi(B+1,nelem-1);
       }
}

void print_path(int *X, int indice, int righe) {
    if(indice < righe)
     {
        cout << '(' << indice << ',' << X[indice] << ')' << ' ';
        print_path(X,indice+1,righe);
     }
    cout << endl;

}

bool mossa(bool (*Palude)[8], int riga, int colonna, int dim, int *C)
{

    if (colonna < 0 || colonna >= 8)
      return false;
    if (riga == dim-1)
      if (Palude[riga][colonna])
        {
         C[riga]=colonna;
         return true;
        }
      else
         return false;
   else
      {
         if (Palude[riga][colonna])
          {
           if (mossa(Palude, riga + 1, colonna - 1, dim, C) ||
           mossa(Palude, riga + 1, colonna, dim, C) ||
           mossa(Palude, riga + 1, colonna + 1, dim, C))
              {
                C[riga] = colonna;
                return true;
              }
          }
      }

    return false;
}

bool partenza(bool (*P)[8], int colonna, int dim, int*Path)
{
 if(colonna==8)
  return false;
 if(P[0][colonna])
   if(mossa(P,0,colonna,dim,Path))
      return true;
```

```
   return partenza(P, colonna+1, dim, Path);
}



int main()
{
    bool Palude[8][8];
    leggi(*Palude, 64);
    int Path[8] = {};
    bool ok;

    ok = partenza(Palude,0, 8, Path);

    if (ok)
        print_path(Path, 0, 8);
     else
        cout << "nessun cammino" << endl;
}
```

Correttezza di partenza e di mossa:

PRE_partenza= Palude[dim][8] definita, Path ha dim elementi e colonna<=8

POST_partenza=restituisce true sse da uno degli elementi di P[0][colonna..7] esiste un cammino dalla riga 0 alla 7 secondo le regole dei cammini)&&(se restituisce true alla Path contiene il cammino più a sinistra)

PRE_mossa= (Palude[dim][8] definita) &&(0<=riga<=dim-1)

POST_mossa= (restituisce true sse da P[riga][colonna] parte un cammino fino alla riga 7) && (se restituisce true, allora Path[riga..7] contiene il cammino più a sinsitra)

Correttezza di mossa:

Caso base: si deve osservare che quando si considera il caso ii, il caso i non si verificato e quando si esamina il passo ricorsivo, entrambi i casi base non si sono verificati,

i) colonna non è tra 0 e 7 quindi P[riga][colonna] non è un elemento di riga per cui non ci può essere cammino da quell'elemento. Quindi la risposta false soddisfa la POST_mossa
ii) se riga=dim-1, siamo sull'ultima riga, se P[riga][colonna]=true allora Path[dim-1]=colonna è il cammino da P[dim-1][colonna] a se stessa, quindi il programma soddisfa POST_mossa e anche se P[riga][colonna]=false, restituire false soddisfa POST_mossa.

Caso ricorsivo: riga non è dim-1, quindi vengono eseguite 3 invocazioni ricorsive che corrispondono ai 3 possibili modi di iniziare un cammino che parte da P[riga][colonna]. La valutazione short-cut del C++ garantisce che le 3 invocazioni sono eseguite nell'ordine che esse hanno nell'istruzione e che non appena una invocazione restituisce true, le altre non vengono eseguite. Visto che riga < dim-1, la PRE_mossa_Ric delle 3 invocazioni è vera e quindi, per ipotesi induttiva, possiamo assumere che sia vera anche la POST_mossa_Ric e in particolare che la risposta true/false è corretta.

Ci sono 2 casi da considerare:

i) il condizionale è falso: visto che abbiamo considerato tutte e 3 le strade possibili, per le POST_mossa_Ric, sappiamo che con c'è cammino da P[riga][colonna] alla riga dim-1. Mossa restituisce false e questo soddisfa POST_mossa;

ii) il condizionale è vero: consideriamo l'invocazione ricorsiva che ha restituito true. Assumiamo che corrisponda al passo verso colonna-1 (le altre mosse si trattano allo stesso modo). Abbiamo visto che vale la PRE_mossa_Ric di quell'invocazione, per cui possiamo usare la sua POST_mossa_ric che ci dice che c'è un cammino da P[riga+1][colonna-1] fino alla riga dim-1 e che il cammino più a sinistra è in Path[riga+1..dim-1]. Ovviamente, P[riga+1][colonna-1] è raggiungibile da P[riga][colonna] per cui c'è anche un cammino da P[riga][colonna] alla riga dim-1 e il cammino più a sinistra lo otteniamo con Path[riga]=colonna. Questo cammino è quello più a sinistra per l'ordine delle 3 invocazioni ricorsive e per POST_ric. Infine restituire true soddisfa completamente POST_mossa.

<u>Correttezza di partenza:</u>

Caso base:

i) colonna = 8, quindi non ho elementi di P[0] da considerare, per cui restituire false soddisfa POST_partenza

Passo ricorsivo:

a) P[0][colonna] è false: non ci può esserci cammino da lì per cui, se c'è un cammino è nella parte [colonna+1..7] della riga 0 e infatti la nostra funzione invoca partenza(P, colonna+1, dim, Path). Visto che il caso (i) non si applica, colonna+1 <=8 per cui vale la PRE_partenza_Ric e quindi la POST_partenza_Ric ci garantisce che l'invocazione ricorsiva ci dà la risposta giusta per gli elementi [colonna+1..7] della riga 0, ma viste le cose vere a questo punto, questa stessa risposta è giusta per tutti gli elementi [colonna..7] della riga 0.
b) non vale (a) quindi P[0][colonna] è true. Viene invocata mossa a partire da questa posizione. E' facile verificare che PRE_mossa_Ric(*) è vera: infatti riga =0. Quindi assumiamocle al ritorno dall'invocazione valga POST_mossa_Ric. Ci sono 2 casi:
   1. se mossa restituisce false la POST_mossa_Ric ci garantisce che non c'è cammino da P[0][colonna], quindi andiamo all'invocazione ricorsiva di partenza con colonna+1 per cercare soluzioni nel resto della riga 0. Il ragionamento da usare qui è lo stesso di (a).
   2. se mossa restituisce true, la POST_mossa_Ric garantisce che Path[0..dim-1] contiene il cammino più a sinistra dalla prima all'ultima riga. Per cui, restituendo true, soddisfiamo completamente POST_partenza. Il cammino restituito in Path è quello più a sinistra per 2 motivi:
      -- esaminiamo la riga 0 da sinistra a destra e interrompiamo l'esecuzione non appena troviamo un cammino, quindi il cammino inizia il prima possibile;
      --POST_mossa_Ric ci garantisce che per il dato punto di inizio venga messo in Path il cammino più a sinistra.

(*) ho usato PRE_mossa_Ric e POST_mossa_Ric per indicare la PRE_mossa e POST_mossa istanziate ai parametri attuali dell'invocazione di mossa eseguita da partenza.