משימה 1. מונופול

דני מאוד אוהבת לשחק מונופול. אבל המשחק מאוד ארוך – בין 5 ל-7 שעות. לכן, **ד**ני מתחילה לחשוב על שינויים לחוקים הקלאסיים. השחקנים במונופול בדרך כלל נעים בכיוון אחד ממשבצת למשבצת ולאחר זמן מה חוזרים להתחלה, וממשיכים כך שוב ושוב. בגרסה החדשה התנועה תהיה עדיין ממשבצת למשבצת, אבל מהמשבצת הנוכחית יכולים להיות כמה אפשרויות למהלך הבא. **ד**ני רוצה למצוא חיבורים מכוונים בין המשבצות, כך שהשחקן כך ששחקן לא יוכל לחזור למשבצת שהוא היה בה בעבר, לא משנה איך הוא נע (כמובן בהתאם לחוקים). בדרך זאת המשחק יהיה קצר יותר.

היא התחילה להכין את הלוח החדש – היא בחרה את מספר המשבצות להיות N (המשבצות ממוספרות מ-1 עד N) והיא הכינה רשימה של M חיבורים (לכל חיבור יש כיוון ואין שום חיבור שמחבר משבצת לעצמה). אם משבצת i מחוברת למשבצת j, אזי אין חיבור ישיר בכיוון ההפוך, כלומר מ משבצת j למשבצת i, בנוסף אין עוד חיבור ישיר ממשבצת i למשבצת j. **ד**ני חשבה שהיא מוכנה עם הלוח החדש, אבל פתאום היא שמה לב שהתנאי שהיא רוצה (כשזזים ממשבצת למשבצת, בעזרת החיבורים, אי אפשר לחזור למשבצת שהיו בה בעבר) לא מתקיים ברשימה של החיבורים שלה. בהתחלה היא חשבה להוריד כמה חיבורים ישירים, אבל זה יצריך לכתוב מחדש את הרשימה, שעלולה להיות מאוד ארוכה. לכן, **ד**ני החליטה להפוך את הכיוון של כמה מהחיבורים המכוונים.

המשימה

אתם משחקים מונופול באופן קבוע עם **ד**ני. לכן אתם רוצים לעוזר לה על ידי כתיבת התוכנית **monopoly**, שתגיד לה איזה חיבורים היא צריכה להפוך כך שהתנאי יתקיים. התוכנית צריכה להכיל את הפונקציה *find_reverse* שתקומפל (תהודר) עם התוכנית של הבודק.

פרטי מימוש

הפונקציה *find_reverse* צריכה להיות מהצורה הבאה:

```
std::string find_reverse (int N, int M, int connections[][2]);
```

היא תקרא רק פעם אחת על ידי תוכנית הבודק עם שלושה פרמטרים: N – מספר המשבצות בלוח החדש, M – מספר החיבורים המכוונים ברשימה של **ד**ני, והמערך *connections* בו יש M שורות, כל אחת מכילה שני מספרים x ו-y – משבצות ההתחלה והסוף של החיבור המכוון. הפונקציה צריכה להחזיר מחרוזת בינארית באורך M – בסדר של המערך *connections*, לכל אחד מהחיבורים צריך לשים '1' במחרוזת אם החיבור צריך להתהפך, ו-'0' אחרת. אם יש יותר מפתרון אחד אתם יכולים להחזיר כל פתרון אפשרי.

אתם צריכים להגיש למערכת את הקובץ **monopoly.cpp**, המכיל את המימוש שלכם לפונקציה *find_reverse*. הקובץ יכול להכיל פונקציות וקוד נוסף, הדרושים לתוכנית שלכם, אבל **אסור שיכיל** פונקציית *main*. בנוסף אל תנסו לקרוא מה- standard input או לכתוב ל-standard output!

מגבלות

- ♣ $3 \le N \le 5 \times 10^5$
- ♣ $3 \le M \le 1.5 \times 10^6$

תתי משימות

תת משימה	נקודות	N	M	מגבלות נוספות
1	0	–	–	הדוגמה
2	15	≤ 7	≤ 21	–
3	40	$\le 10^3$	$\le 5 \times 10^3$	–
4	25	$\le 10^5$	$\le 5 \times 10^5$	–
5	20	$\le 5 \times 10^5$	$\le 1.5 \times 10^6$	–

על מנת לקבל ניקוד לתת משימה, הפתרון צריך לעבור את כל הטסטים בתת המשימה.

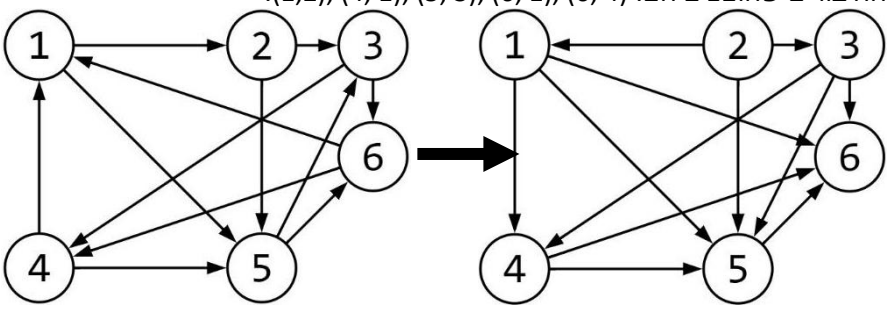
בדיקות מקומיות

על מנת לבדוק מקומית אתם מקבלים את הקובץ **Lgrader.cpp**. אתם צריכים לשים אותו באותה תיקייה של התוכנית שלכם **monopoly.cpp** ועליכם לקמפל (להדר) את הקבצים **Lgrader.cpp** ו-**monopoly.cpp** ביחד. כך אתם תיצרו תוכנית שתבדוק את נכונות הפונקציה שלכם. התוכנית תדרוש את רצף הנתונים הבא מה-standard input :

- בשורה הראשונה: שני מספרים שלמים – מספר המשבצות N ומספר החיבורים M של הלוח החדש.
- בכל אחת מ-M השורות הבאות: שני מספרים שלמים x ו-y - משבצות ההתחלה והסיום של כל אחד מהחיבורים המכוונים.

הפלט של התוכנית יהיה המחרוזת הבינארית שמצאתם.

דוגמה לבדיקה מקומית

קלט	פלט	הסבר
6 12 1 2 1 5 2 3 2 5 3 4 3 6 4 1 4 5 5 3 5 6 6 1 6 4	100000101011	החיבורים שהופכים הם: (1,2), (4, 1), (5, 3), (6, 1), (6, 4).  האיור מעל מראה את המשבצות, ראשית עם החיבורים המכוונים מהרשימה של **ד**ני, ושנית לאחר היפוך הכיוון של החיבורים המתאימים ל-'1' בפלט. בבירור בהתחלה אם נשתמש בחיבורים המכוונים (1, 2), (2, 3), (3, 4) ו-(4, 1) אז אפשר להתחיל ממשבצת 1 ולנוע על חיבורים אלו, ולהגיע חזרה למשבצת. אפשר לראות שהתנאי של **ד**ני מתקיים בלוח השני. שימו לב שיש עוד פתרונות חוקיים: (4, 1), (5, 3), (6, 1), (6, 4) (1, 5), (2, 3), (2, 5), (3, 4), (3, 6), (4, 5), (5, 6)