Úkolem je vytvořit program, který bude počítat umístění bodů v rovině tak, aby vznikly rovnoběžníky.

V rovině jsou zadané 3 body: A, B a C. Každý z bodů je zadaný pomocí svých souřadnic (dvojice desetinných čísel). Program tyto souřadnice přečte ze svého vstupu a určí souřadnice čtvrtého bodu tak, aby vznikl rovnoběžník (případně nějaká ze speciálních variant rovnoběžníku: kosočtverec, obdélník nebo čtverec). Souřadnici čtvrtého bodu program zobrazí na svém výstupu, dále zobrazí informaci o případném speciálním tvaru vznikajícího obrazce (čtverec/obdélník/ kosočtverec/rovnoběžník). Obecně existují 3 možnosti umístění bodu (rovnoběžníky ABA'C, ABCB' a AC'BC), viz ukázka běhu programu níže.

Může se stát že vstupní body leží na přímce, pak zadání nemá žádné řešení. Program tuto situaci musí detekovat a odpovídajícím způsobem zareagovat (viz ukázka běhu programu níže).

Pokud je vstup neplatný, program to musí detekovat a zobrazit chybové hlášení. Chybové hlášení zobrazujte na standardní výstup (ne na chybový výstup). Za chybu považujte:

- nečíselné zadání souřadnic (neplatné desetinné číslo),
- chybějící souřadnice,
- chybějící nebo přebývající oddělovače (souřadnice musí být zadaná v hranatých závorkách, hodnoty x a y musí být oddělené čárkou).

Ukázka práce programu:

Bod A:
[0, 0]
Bod B:
[7, 0]
Bod C:
[3, 2]
A': [10,2], rovnobeznik
B': [-4,2], rovnobeznik
C': [4,-2], rovnobeznik

Bod A:
[0,0]
Bod B:
  [5 , 0   ]
Bod C:

[3,
4
]
A': [8,4], kosoctverec
B': [-2,4], rovnobeznik
C': [2,-4], rovnobeznik

Bod A:
[0,0]
Bod B:
[-3,4]
Bod C:
[4,3]
A': [1,7], ctverec
B': [7,-1], rovnobeznik
C': [-7,1], rovnobeznik

Bod A:
[10.5, 10.5]   [12.5, 10.5][10.5, 15e+0]
Bod B:
Bod C:
A': [12.5,15], obdelnik
B': [8.5,15], rovnobeznik
C': [12.5,6], rovnobeznik

Bod A:
[0, 0]
Bod B:
[3, 3]
Bod C:
[10, 10]
Rovnobezniky nelze sestrojit.

Bod A:
[0, 0]
Bod B:
[2270.242, 0]
Bod C:
[234.08, 2258.142]
A': [2504.322,2258.142], kosoctverec
B': [-2036.162,2258.142], rovnobeznik
C': [2036.162,-2258.142], rovnobeznik

Bod A:
[740.865, 887.560]
Bod B:
[340.090, 1241.872]
Bod C:
[1095.177, 1288.335]
A': [694.402,1642.647], ctverec
B': [1495.952,934.023], rovnobeznik
C': [-14.222,841.097], rovnobeznik

Bod A:
[-306.710, -894.018]
Bod B:
[6369.015, 66159.129]
Bod C:

[6016.590, 62619.258]
Rovnobezniky nelze sestrojit.

Bod A:
[2, 5]
Bod B:
[3, abcd]
Nespravny vstup.

Bod A:
[2, 5]
Bod B:
[3, 4]
Bod C:
[7 9]
Nespravny vstup.

Poznámky:

- Ukázkové běhy zachycují očekávané výpisy Vašeho programu (tučné písmo) a vstupy zadané uživatelem (základní písmo). Zvýraznění tučným písmem je použité pouze zde na stránce zadání, aby byl výpis lépe čitelný. Váš program má za úkol pouze zobrazit text bez zvýrazňování (bez HTML markupu).
- Znak odřádkování (\n) je i za poslední řádkou výstupu (i za případným chybovým hlášením).
- Pro reprezentaci hodnot použijte desetinná čísla typu double. Nepoužívejte typ float, jeho přesnost nemusí být dostatečná.
- Úlohu lze vyřešit bez použití funkcí. Pokud ale správně použijete funkce, bude program přehlednější a bude se snáze ladit.
- Číselné vstupní hodnoty jsou zadávané tak, aby se vešly do rozsahu datového typu double. Referenční řešení si vystačí s číselnými typy double a int.
- Pro načítání vstupu se hodí funkce scanf, podívejte se na konverze "%c" a " %c" (s mezerou před konverzí, najděte si v manuálu rozdíl).
- Při programování si dejte pozor na přesnou podobu výpisů. Výstup Vašeho programu kontroluje stroj, který požaduje přesnou shodu výstupů Vašeho programu s výstupy referenčními. Za chybu je považováno, pokud se výpis liší. I chybějící nebo přebývající mezera/odřádkování je považováno za chybu. Abyste tyto problémy rychle vyloučili, použijte přiložený archiv se sadou vstupních a očekávaných výstupních dat. Podívejte se na videotutoriál (courses → výuková videa), jak testovací data použít a jak testování zautomatizovat.
- Hodnoty souřadnic na výstupu Vašeho programu se porovnávají s referenčními hodnotami. Porovnání toleruje malé rozdíly desetinných čísel (menší než 1 %).

- Váš program bude spouštěn v omezeném testovacím prostředí. Je omezen dobou běhu (limit je vidět v logu referenčního řešení) a dále je omezena i velikost dostupné paměti (ale tato úloha by ani s jedním omezením neměla mít problém). Testovací prostředí dále zakazuje používat některé "nebezpečné funkce" -- funkce pro spouštění programu,

pro práci se sítí, ... Pokud jsou tyto funkce použité, program se nespustí. Možná ve svém programu používáte volání:

```
int main ( int argc, char * argv [] )
{

  ...

  system ( "pause" ); /* aby se nezavrelo okno programu */
  return 0;
}
```

Toto nebude v testovacím prostředí fungovat - je zakázáno spouštění jiného programu. (I pokud by se program spustil, byl by odmítnut. Nebyl by totiž nikdo, kdo by pauzu "odmáčkl", program by čekal věčně a překročil by tak maximální dobu běhu.) Pokud tedy chcete zachovat pauzu pro testování na Vašem počítači a zároveň chcete mít jistotu, že program poběží správně, použijte následující trik:

```
int main ( int argc, char * argv [] )
{

  ...

#ifndef __PROGTEST__
  system ( "pause" ); /* toto progtest "nevidi" */
#endif /* __PROGTEST__ */
  return 0;
}
```