майката на Виктор го събужда. На третия ред числата са три – необходимия брой минути за сутрешния тоалет на Виктор, необходимия брой минути за закуска и необходимия брой минути за подготовка на чантата. Всички минути във входните данни са цели числа от 0 до 59.

Изход

Програмата трябва да изведе `Yes`, ако при така зададените времена Виктор ще закъснее за училище и `No`, ако няма да закъснее.

ПРИМЕР 1	**ПРИМЕР 2**
Вход	**Вход**
8 0 6 30 7 10 2	8 0 7 00 10 7 2
Изход	**Изход**
No	Yes

Решение

От времето на започване на учебните занятия (час `h` и минути `m`) изваждаме 5 минути, за да получим в колко часа най-късно Виктор трябва да бъде в училище. Правим това внимателно, защото `m` може да е по-малко от 5. След това пресмятаме колко време общо е необходимо на Виктор за сутрешен тоалет, закуска, приготвяне на чантата и път до училище (`ob`). Към това време добавяме и минутите на ставане (`wm`). След това преобразуваме полученото време на пристигане в училище в нормален вид (час `lh=wh+ob/60` и минути `lm=ob%60`). Това време сравняваме с времето, в което най-късно Виктор трябва да е на училище, за да не закъснее и извеждаме съответно `Yes` или `No`. Първо сравняваме двата часа (`lh` и `h`) и само когато те са равни, сравняваме и минутите (`lm` и `m`).

Вариант на C/C++

```
#include <iostream>
using namespace std;
int main()
{  unsigned short int h,m,wh,wm,st,mz,tb,ob,p,lh,lm;
   cin>>h>>m; cin>>wh>>wm; cin>>st>>mz>>tb;
   if (m>=5) m=m-5; else {h=h-1; m=m+60-5;}
   p=4*(st+tb); ob=st+mz+tb+p+wm;
   lh=wh+ob/60; lm=ob%60;
   if (lh<h) cout<<"No"<<endl;
   else if ((lh==h) && (lm<=m)) cout<<"No"<<endl;
        else cout<<"Yes"<<endl;
   return 0;
}
```

Вариант на Pascal

```
program Exact;
var
   h,m,wh,wm,st,mz,tb,ob,p,lh,lm:integer;
begin
   readln(h,m); readln(wh,wm); readln(st,mz,tb);
```



```
   if m>=5 then m:=m-5
   else
   begin
      h:=h-1; m:=m+60-5;
   end;
   p:=4*(st+tb);
   ob:=st+mz+tb+p+wm;
   lh:=wh+ob div 60;
   lm:=ob mod 60;
   if lh<h then writeln('No')
   else
      if (lh=h) and (lm<=m) then writeln('No')
       else writeln('Yes');
end.
```

Задача Е2. ТЕКСТ, автор Бисерка Йовчева

Виктор току що е научил, че една дума се нарича ***палиндром*** когато се чете по един и същ начин отляво надясно и отдясно наляво. Такива са думите “капак”, “невен”, “боб” и др. Съставете програма **TEXT**, която проверява дали зададена дума е палиндром.

Вход

Програмата трябва да прочете зададената дума от клавиатурата. Думата ще бъде съставена от седем букви на латинската азбука (както малки така и главни, като съответните малка и главна буква считаме за еднакви).

Изход

Ако въведената дума е палиндром, програмата трябва да я изведе с главни букви, а ако не е палиндром – с малки.

ПРИМЕР 1	**ПРИМЕР 2**
Вход	**Вход**
abcDcBa	TaratoR
Изход	**Изход**
ABCDCBA	tarator

Решение

За да решим задачата първо трябва да прочетем думата, буква по буква, и да проверим дали е палиндром. Една седембуквена дума е палиндром, ако са еднакви (включително голяма и малка от един вид) следните двойки букви: първа и седма, втора и шеста, трета и пета. Ако думата е палиндром, ще преобразуваме всички малки букви в нея до съответните главни, а ако не е палиндром – всички главни в малки. Една малка буква се превръща в главна като към стойността и се добави `'A'-'a'`, а за обратното преобразуване – като се добави `'a'-'A'`. Тъй като и в двата случая към буквата се прибавя една и съща стойност, добре е да я пресметнем еднократно в началото. Преобразуването на буквите се осъществява за всяка една буква поотделно. Предлагаме две различни решения, в зависимост от това дали учениците познават масивите от знаци или не. И в двата случая използваме цялата

променливата l като Булева, за да помним дали думата е палиндром, а в променливата d запомняме стойността на `'A'-'a'` или `'a'-'A'`.

Вариант 1 (без използване на масив): В този случай буквите на думата съхраняваме в седем отделни знакови променливи `c1,c2,c3,c4,c5,c6, c7` – по една за всяка от буквите.

```
int main()
{
   char c1, c2, c3, c4, c5, c6, c7;
   int d,l;
   cin.get(c1);cin.get(c2);
   cin.get(c3);cin.get(c4);
   cin.get(c5);cin.get(c6);
   cin.get(c7);
   l=c1==c7||c1==c7+'A'-'a'||c7==c1+'A'-'a';
   l=l&&(c2==c6||c2==c6+'A'-'a'||c6==c2+'A'-'a');
   l=l&&(c3==c5||c3==c5+'A'-'a'||c5==c3+'A'-'a');
   d=l?('A'-'a'):('a'-'A');
   if(l&&c1>='a'&&c1<='z'||!l&&c1>='A'&&c1<='Z')c1+=d;
   if(l&&c2>='a'&&c2<='z'||!l&&c2>='A'&&c2<='Z')c2+=d;
   if(l&&c3>='a'&&c3<='z'||!l&&c3>='A'&&c3<='Z')c3+=d;
   if(l&&c4>='a'&&c4<='z'||!l&&c4>='A'&&c4<='Z')c4+=d;
   if(l&&c5>='a'&&c5<='z'||!l&&c5>='A'&&c5<='Z')c5+=d;
   if(l&&c6>='a'&&c6<='z'||!l&&c6>='A'&&c6<='Z')c6+=d;
   if(l&&c7>='a'&&c7<='z'||!l&&c7>='A'&&c7<='Z')c7+=d;
   cout<<c1<<c2<<c3<<c4<<c5<<c6<<c7<<endl;
   return 0;
}
```

Вариант 2 (с използване на масив)**:** В този случай буквите на думата съхраняваме в елементите на масива `c[7]`.

```
#include <iostream>
using namespace std;
int main()
{
   char c[7]; int d,l,i;
   cin>>c;
   l=c[0]==c[6]||c[0]==c[6]+'A'-'a'||
     c[6]==c[0]+'A'-'a';
   l=l&&(c[1]==c[5]||c[1]==c[5]+'A'-'a'||
     c[5]==c[1]+'A'-'a');
   l=l&&(c[2]==c[4]||c[2]==c[4]+'A'-'a'||
     c[4]==c[2]+'A'-'a');
   d=l?('A'-'a'):('a'-'A');
   for(i=0;i<7;i++)
       if(l&&c[i]>='a'&&c[i]<='z'||!l&&c[i]>='A'&&
             c[i]<='Z') c[i]+=d;
   cout<<c<<endl ;
}
```


Задача Е3. ЛИНИЙКА, автор Бистра Танева

Виктор има N летвички ($2 \le N \le 100$) и се пита какви са дължините на най-дългата и най-късата летвичка. За целта решил да използва линийка с дължина 100 см, на която са нанесени деления през 1 см. Оказало се, че дължините на всички летвички са цели числа не надминаващи 100. Виктор обича да си създава трудности и вместо да измерва дължините, започвайки от началото на линийката, поставял единия край на летвичката на произволно деление на линийката и си записвал мястото на двата й края. При това даже не подреждал двете числа по един и същ начин. Сега ще му трябва програма **RULER**, която да намери най-голямата и най-малката дължина.

Вход

Програмата трябва да въвежда данните от клавиатурата. На първия ред ще бъде зададено числото N, а на всеки от следващите N реда по една двойка цели числа, разделени с интервал – мястото на краищата на една от летвичките върху линийката.

Изход

Програмата трябва да изведе на един ред, разделени с интервал, дължините на най-голямата и най-малката летвичка.

ПРИМЕР

Вход	**Изход**
5 3 9 8 12 15 6 10 19 0 25	25 4

Решение

Преди да започнем обработката, задаваме като стойност на променливата `max` най-малката възможна дължина 0, а на променливата `min` – най-голямата възможна дължина 100. В променливите `a1` и `a2` въвеждаме двойката числа, показващи местата на краищата на летвичката върху линийката. Тъй като не е указано кой край е първи (този намиращ се по-близо до началото на линийката или по-далечния), проверяваме кое от въведените числа по-голямо. От него изваждаме другото число, за да намерим дължината на летвичката, която запомняме в променливата `a`. За пресметнатата дължина проверяваме дали не е по-малка от намерения до момента минимум или по-голяма от намерения до момента максимум и, ако се налага, заменяме старата стойност с новата. Дължината може да намерим и като пресметнем абсолютната стойност на разликата `a1-a2` (с функцията `abs` или като умножим с –1 разликата, ако е отрицателно число) .

I начин: Без да се използва вградената функция `abs`.

```
#include <iostream>
using namespace std;
int main()
{  unsigned short int a1,a2,max=0,min=100,n;
```