Problem F

Problem F
Walking the matrix

Your task is to create a program processing input matrices containing decimal digits to find the biggest possible number divisible by three. Every matrix is a square and it contains 25 decimal digits - 5 rows 5 digits each. The resulting numbers are constructed in the following way:

* you must start in the upper left corner of the matrix - the digit in this corner is the leftmost, most significant digit of the number being constructed,
* you must end your route in the lower right corner of the matrix - the digit in this corner is the rightmost, least significant digit of the number being constructed,
* all other elements of the matrix may be visited only once or not visited at all,
* the resulting number is constructed from left to right, by attaching consecutive digits to its right side,
* there are only four possible moves: to the left, to the right, up and down (provided you obey other rules, of course),
* you must not pass the boundaries of the matrix; this means that if you for example reach the rightmost element in a row you cannot pass to the leftmost element of this row in one move.

Input
The input is as follows:

* in the first line there is the number of matrices in the input,
* in the next lines there are rows of consecutive matrices, one row in a line, the digits are separated with spaces and/or tabs, the line ends with the end-of-line character,
* the last line of the input ends with the end-of-file character.

Output
The output should contain:

* as many lines of answers as the number of matrices in the input,
* every line should contain one solution i.e. the biggest possible number for the corresponding matrix or - if for any reason it is not possible to construct one, the number 0,
* all lines should end with the end-of-line character.

EXAMPLE

Input
For example, given the following input (three matrices):

```
3
1 2 3 4 5
6 7 8 9 0
1 2 3 4 5
6 7 8 9 0
1 2 3 4 5
```

```
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
3 3 3 3 3
1 1 1 1 1
1 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

Output
We obtain the following answers (for each input matrix one answer):

```
167238945054387216123 4905
000333113000000000000 31
0
```

Solution

Test

```
input        5
         1 2 3 4 5
         6 7 8 9 0
         1 2 3 4 5
         6 7 8 9 0
         1 2 3 4 5
         0 0 0 0 0
         0 0 0 0 0
         0 0 0 0 0
         3 3 3 3 3
         1 1 1 1 1
         5 5 5 5 5
         4 4 4 4 4
         3 3 3 3 3
         2 2 2 2 2
         1 1 1 1 1
         1 0 0 0 0
         0 0 0 0 0
         0 0 0 0 0
         0 0 0 0 0
         0 0 0 0 1
         2 3 9 8 9
         1 2 3 4 0
         0 9 0 2 1
         2 3 4 6 1
         0 9 8 1 2


output

         167238945054387216123 4905
         000333113000000000000 31
         555554444433333222221 1111
         0
         23989043210939840211612
```

Listing

Listing

```
#include
#include

char szCTask[26], szSol[26], szBestSoFar[26];
unsigned short usVisited[25], usSolLen;

int div3(const char szNum[])    /* checks divisibility by 3 */
{
  unsigned uSum, uI;
  uSum = uI = 0;
  while(szNum[uI] != '\0')uSum += szNum[uI++]-'0';
  /* sum all digits */
  return(!(uSum % 3));
}
/* compares two strings/numbers */
int gt(const char szs1[], const char szs2[])
{
  unsigned uL1, uL2, uI;
  char szS1[26], szS2[26];
  /* remove all trailing zeroes */
  uL1 = 0; while(szs1[uL1]=='0')uL1++;
  strcpy(szS1,&szs1;[uL1]);
  uL2 = 0; while(szs2[uL2]=='0')uL2++; /* as above */
  strcpy(szS2,&szs2;[uL2]);
  uL1 = strlen(szS1);
  uL2 = strlen(szS2);
  if(uL1 != uL2)
    if(uL1 > uL2)return(1); /* longer is bigger */
    else return(0);         /* shorter is smaller */
  else
    /* compare digits left to right */
    for(uI = 0;szS1[uI]!='\0';uI++)
    {
      if(szS1[uI]>szS2[uI])return(1);
      if(szS1[uI] 4)
      if(usVisited[usI-5]==0)findSolution(usI-5); /* move up */
    if(usI < 20)
      if(usVisited[usI+5]==0)findSolution(usI+5); /*move down*/
  }
  usVisited[usI] = 0;
  szSol[--usSolLen]='\0';
}

void getTask(void)      /* reads task from console */
{
  unsigned short usCElem, usBuf;
  for(usCElem=0;usCElem<25;usCElem++)
  {
    scanf("%hu",&usBuf;);
    szCTask[usCElem]=usBuf+'0';
    usVisited[usCElem] = 0;
  }
  szCTask[25]=szBestSoFar[0]='\0';
  usSolLen = 0;
}

int main (void)
{
  unsigned short usTasksNo, usCTask;
  scanf("%hu",&usTasksNo;);
  for(usCTask=0;usCTask < usTasksNo;usCTask++)
  {
    getTask();
    findSolution(0);
    if(szBestSoFar[0]!='\0')puts(szBestSoFar);
    else puts("0");
  }
  return(0);
```

```
}
```


 zwir, *wierzej*, Mon Oct 28 23:01:26 MET DST 1996