Table of Contents

INTRODUCTION:

What special patterns can be found among the valid addition expression of the form A+B=C, such that each digit from one (1) to nine (9) is used exactly once in A, B, or C? An example of such a problem is:

124	162
659	387
783	549

The problem seems simple, but may questions arise. How many pairs of addends will give a valid solution? How can the problem be expanded? What area of math does the problem reside? We chose this problem because it is easy to understand what is required, but it has real depth in trying to figure out why.

HYPOTHESIS:

If such sums are found, patterns will exist among them.

METHOD:

1. By trial and error, find valid sums.
2. Use mathematics to discover special properties and prove as theorems.
3. Use a computer program (see Appendix A for the program) to find all the program between 123 and 987 that would form such sums (the number 123 and 987 are chosen as the limit numbers because they are respectively the lowest and the highest numbers that are allowed according to the problem).
4. Cast out any duplicate sums due to communitivity.
5. Look for patterns and form groups that have similar answers.
6. Find all commonalties that all such sums have.
7. Repeat process for base 13 and base 16.

DESCRIPTION

There are, in fact, patterns in the addition problems. The problems have to be arranged in the format ABC+DEF=GHI, because it is the only possible way to include all numbers one (1) through nine (9) exactly once. The other forms are possible because they do not allow the use of the numbers one (1) through nine (9) only once. They require multiple uses of certain numbers and no uses of other numbers, causing the format to be invalid.

1)	2)	3)
ABCD	ABC	ABC
___ E	_ DE	DEF
FGHI	FGHI	GHI

The reason that combination number one (1) cannot work is because, in order to work, A must equal F, causing the answer to be invalid, This is because any number one (1) through nine (9) can only be used once to make the problem valid. An example follows:

1725
__9
1734

Problem number two (2) cannot work because there would be a zero in the sum and that violates the problem criteria. There would be a zero in the sum because the largest number in the first addend could be a nine (9). This would result in a ten (10) in the sum, causing the problem criteria to be violated because only the numbers one (1) through nine (9) may be used in the addition problem. An example follows:

972
_54
1026

There are many further applications of this type of problem. One can try to fashion the same type of problem using the numbers one (1) through six (6) in a two (2) by two (2) fashion, using the format:

A B
C D
E F

One may also be able to use the base thirteen (13) in a four (4) by four (4) format:

A B C D
E F G H
I J K L

Yet another possible problem may involve the use of hexadecimal (base sixteen) in the Format:

A B C D E
F G H I J
K L M N O

These problems must use the two (2) by two (2) format, the four (4) by four (4) format, or the five (5) by five (5) format because, as seen previously, there must be an equal amount of numbers in both of the addends and in the sum. These types of problems may or may not work out, but if they do, one knows how to start to figure them out.

Another pattern that is noticed is that there must be one carry per valid sum. This is the first theorem.

The first theorem states: For every valid number set, there must be exactly one carry. A proof of theorem two (2) follows. If there are no carries in the problem, then a nine (9) must occur in the sum. This is so because any number one (1) through eight (8) added to nine (9) will yield a number that has two digits, which will result in a carry. All possible valid combinations of numbers one (1) through eight (8) that will yield a nine are following. These numbers will be used to prove theorem two (2):

8+1=9 7+2=9 6+3=9 5+4=9

125 This sum is not valid because there are two (2) eights and no sevens,
843 violating the criteria given in the problem.
969

647 This sum is not valid because there are two (2) nines and no eights.
312
959

716 This sum is not valid because there are two (2) nines and no eights.
243
959

572 This sum is not valid because there are two (2) fives and no sixes.
413
985

The previous examples show that there must be a carry in order to preserve the rules of the problem, that all numbers one (1) through nine (9) are used exactly once in a valid sum.

Another pattern that the experimenter found is that there must be a carry in either column one or column two (going from left to right), but not in both columns. This must occur in every valid sum. This is called theorem two (2).

Theorem two (2) states: For every valid sum, there must be a carry in either the first or second column (going from left to right), but not in both columns and not in the third column.

An example of this is as follows (where problem number one (1) is a first column carry, problem number two (2) is a second column carry, problem number three (3) is an invalid attempt to carry on both the first and second columns, and problem number four (4) is an invalid attempt to carry in the third column). In the last two (2) example problems, the answers are not valid, according to the criteria given in the problem:

3 2 1	3 2 1	3 2 1	3 2 1
1) 246	2) 376	3) 168	4) 618
735	542	359	539
981	918	527	1157

According to theorem one (1) a carry must occur. However, a carry cannot occur in both the first and the second columns because it results in repeating numbers and that violates the rules of the problem (shown in number three (3). Also, a carry cannot occur in the third column because only nine (9) numbers are allowed to be used exactly once and a third column carry would violate that rule as shown above in number four (4). Therefore, a carry must occur in either the first or second column.

Another pattern that is noticed is that some of the sums have four (4) different possible combinations of addends while others have eight (8) different possible combinations of addends (refer to the computer generated chart for every possible combinatiuon of addends in Appendices B,C, or D). There will always be four (4) or eight (8) possible addend combinations, due to the way the numbers are arranged. To find the combinations of four (4), it is required that certain numbers in two (2) addends be rearranged. The addends can be rearranged four (4) different ways causing four (4) different combinations of numbers that lead to the same sums. Thus, one (1) valid solution will really yield four (4) valid solutions.

The first combination of numbers is the original number set (the original valid solution). The second combination occurs when the addend numbers in the first column are vertically reversed (C/F goes to F/C). The third combination occurs when the numbers in the second column are vertically reversed (B/E goes to E/B). Finally, the fourth combination occurs when the numbers in the third column are vertically reversed (A/D goes to D/A). These rearrangements of numbers results in four (4) different number combinations:

3 2 1	3 2 1	3 2 1	3 2 1
1) ABC	2) ABF	3) AEC	4) DBC
DEF	DEC	DBF	AEF
GHI	GHI	GHI	GHI

1) 138	2) 134	3) 158	4) 638
654	658	634	154
792	792	792	792

There are also occurrences of eight (8) possible combinations of numbers. These different combinations are caused by rearranging the numbers in the two (2) addends.

Another pattern is that, for both the sums with four (4) possible combinations and eight (8) possible combinations, a reversal of the entire first and third column (including the part of the sum) will allow a new number to be formed. However, it must be noted that a carry still has to occur in only the first and second columns and not at all in the third column. This is labeled theorem three (3). 4

Theorem three (3) states: A reversal of the entire first and third columns will result in a new sum with different addends. A carry is only allowed to occur in the first and second columns. However, the first two (2) columns or the last two (2) columns (depending on how the numbers are reversed) must always remain together to prevent a carry in the third column. An example follows (where problem number one (1) is the original and problem number two (2) is the new problem):

Example 1:

3 2 1	3 2 1	3 2 1	3 2 1
1) 214	1) 219	1) 254	1) 614
659	654	619	259
873	873	873	873

2 1 3	2 1 3	2 1 3	2 1 3
2) 142	2) 146	2) 192	2) 542
596	592	546	196
738	738	738	738

Example 2:

3 2 1	3 2 1	3 2 1	3 2 1
1) 218	1) 219	1) 248	1) 318
349	348	319	319
567	567	567	567

2 1 3	2 1 3	2 1 3	2 1 3
1) 128	1) 129	1) 138	1) 428
439	438	429	139
567	567	567	567

2 1 3	2 1 3	2 1 3	2 1 3
2) 182	2) 183	2) 192	2) 482
493	492	483	193
675	675	675	675

3 2 1	3 2 1	3 2 1	3 2 1
2) 281	2) 284	2) 291	2) 381
394	391	394	294
675	675	675	675

In the first example, the third column moved to the back (going from columns 3, 2, 1 to columns 2, 1, 3), crating an entire new number. The same is true for example two (2). The column in the front moved to the end of the sum, creating an entire new number with new combinations of addends.

There are carries in the number one (1) columns of each example problem. The first example has four (4) possible combinations for problem number one (1) and four (4) possible combinations for problem number two (2). The second example has eight (8) possible combinations for problem number one (1) and eight (8) possible combinations for problem number two (2)

Another pattern that has been noticed is that, in the sums with eight (8) possible addend combinations, all sums except for those that contain four (4) have a carry in the same column throughout all eight (8) possible combinations. An
Example follows:

1) 235	2) 245	3) 236	4) 246	5) 324	6) 354
746	736	745	735	657	627
981	981	981	981	981	981

7) 327	8) 357
654	624
981	981

1)	2)	3)	4)	5)	6)
216	236	218	238	271	281
738	718	736	716	683	673
954	954	954	954	954	954

7)	8)
273	283
681	671
954	954

The first is an example of a carry in the first column that is continuous through all eight (8) possible combinations of numbers. There are combinations of eight that have a carry in the second column through all possible combinations. The second example has a carry in the first column for the first four (4) combinations and a carry in the second column for the last four (4) combinations. This is because the sum contains a four.

These documented patterns are just a few of the patterns that exist among the results given. There are many more waiting to be found.

CODE

The first C program utilizes a brute force algorithm; first, an array, called digits[], records all the possible combinations of 3 digit numbers where each digit is a unique digit between 1_9 as in the following snippet of code:

```
for(a=1;a<10;a++)
{
 for(b=1;b<10;b++)
 {
  for(c=1;c<10;c++)
  {
   if(a!=b && a!=c && b!=c)
   {
    unique_num=(a*100) + (b*10) + c;
    digits[unique_count]=unique_num;
    unique_count++;
   }
  }
 }
}
```

Next, a temporary variable, unique_sum, is set to the sum of each possible combination of unique 3-digit factors which have been generated and stored in the digits[] array:

```
for(i=0;i<unique_count;i++)
{
 for(j=0;j<unique_count;j++)
 {
  if(digits[i]!=digits[j])
  {
   unique_sum=digits[i]+digits[j];
   a=digits[i]/100;
   b=(digits[i]/10)%10;
   c=digits[i]%10;
   d=digits[j]/100;
   e=(digits[j]/10)%10;
   f=digits[j]%10;
   ones=unique_sum%10;
   tens=(unique_sum/10)%10;
   hundreds=unique_sum/100;
```

Finally, a conditional statement tests whether each sum and it's corresponding factors meets the conditions of the original problem, and the resulting data is written to a file:

```
if(a<d && unique_sum<1000 && a!=b && a!=c && a!=d && a!=e
&& a!=f && a!=ones && a!=tens && a!=hundreds && b!=c && b!=d
&& b!=e && b!=f && b!=ones && b!=tens && b!=hundreds && c!=d
&& c!=e && c!=f && c!=ones && c!=tens && c!=hundreds && d!=e
&& d!=f && d!=ones && d!=tens && d!=hundreds && e!=f && e!=ones
&& e!=tens && e!=hundreds&& f!=ones && f!=tens && f!=hundreds
&& ones!=tens && ones!=hundreds && tens!=hundreds && ones!=0
&& tens!=0 && hundreds!=0)
   {
    fprintf(fp,"\t %d %d %d ",a,b,c);
    fprintf(fp,"\t\t%d %d %d",d,e,f);
    fprintf(fp,"\t\t%d %d %d\n",hundreds,tens,ones);
    sum_count++;
   }
```

We finally were forced to give up on programming the project in C. In order to make our algorithm more efficient? We had difficulty in changing integers to strings. String variables are necessary since number bases larger than 10 use letters to represent digits, and it

would make our checking to see if our sums were valid or not more streamline. The design of the new algorithm was essentially thus:

Example:

> Take three numbers such as 124, 659, and 124 + 659 = 783.
> Convert the integers to string variables: “124”, “659”, and “783”.
> Concatenate them into one large string: “124659783”.
> Sort the elements of the string in ascending order, “123456789”,
> And compare the result to “123456789”. If a match is found, then
> The sum is deemed valid.

Notice that 124, 659, and 783 formed a valid sum. Consider 124, 569, and 683. The algorithm would convert this to “123456689”, which is not the same as “123456789” and therefore would not be valid.

Since we were testing using strings, this allowed us to adapt the program to bases 13 and 16. The programs were then again refined to skip over large sections of invalid numbers by taking advantage of early matching. If a was in the 100’s then b must be in 200’s or larger. So we could b at 234 instead. This became very important, if we were to manage the larger bases.

RESULTS

Five computer programs were developed. For base 10 a total of 168 possible valid combinations were found. The results are listed in the appendix. Partial listing were done for the larger bases, since the number of sums to compare was large. The upper limit is 13^{10} for base 13, and 16^{12} for base 16. The result would be lengthy computer time and large data files. By examining a partial table we were able to get an idea of what was happening. An interesting side discovery was that resulting sums were relatively constant.

Base	Sum
10	18
13	33, 39
16	60

This was probably our most astonishing discovery! Why this occurs and what causes it still remains a mystery.

CONCLUSIONS

The experiment was conducted in a very efficient manner as the experimenters quickly found certain characteristics of all sums, making it easier to find more valid expressions. The hypothesis states that there will be patterns that exist among the various sums. There are in fact many patterns that exist among the various sums and there may be more that have yet to be discovered. The development of a more streamline algorithm was definitely accomplished.

RECOMMENDATIONS

The patterns that remain unsolved need further research. The sum of the digits of the two addends totaling to a constant in every valid sum is a bit unnerving. Furthermore, we need to translate the pascal programs into C or C^{++} so that we can take advantage of a supercomputer. This would enhance speed and accommodate large data files.

ACKNOWLEDGEMENTS

Dr. Curtis Barefoot
Gina Fisk
David Kratzer
Eric Ovaska

REFERENCES

Dover, Gandmier. Mathematics, Magic and Mystery. Martin Press, 1956.
Edward, Kasner. Mathematics and the Imagination. Schusterand Simon, 1967.
Kline, Morris. Mathematics and the Search for Knowledge. Oxford University Press, 1985

Http://www.seanet.com/~ksbrown/htt 06/20/2001

Http://www.ceismc.gatech.edu/busyt/math.html 06/26/2001

http://www.loc.gov/06/26/2001

Appendices

Appendix A: First C Program

```
//numberpuzzle_v1.c
//Author:  Cherie O'Dell
//Date:  June 27, 2001
//This program generates a list of all possible problems with 3 digit sums, where the factors
//are exactly 3 digits, and every digit in the problem is unique; and where the numbers 1_9
//appear in the problem exactly once.
//Note: there are bugs in this code; invalid data is generated along with valid data.
#include <stdlib.h>
#include <stdio.h>

FILE *fp;

main()
{
//a, b, & c are the 3 digits of the first factor of each generated problem
 //d, e, & f are the 3 digits of the second factor of each generated problem
 //hundreds, tens, & ones are the 3 digits of each generated sum
 //i & j are counters
 //unique_count records the number of unique 3 digit numbers possible
 //sum_count records the number of possible sums which meet the conditions of the problem
 //digits[] holds all possible unique 3 digit numbers
 //unique_sum is a temporary variable holding a unique sum

 int a, b, c, d, e, f, i, j, hundreds, ones, tens, unique_count, sum_count, digits[1000];
 sum_count=unique_count=0;

 fp = fopen("3dignums.txt", "wt");

 for(i=0;i<1000;i++)
 {
  abc[i]=0;
  def[i]=0;
 }

 for(a=0;a<10;a++)
 {
  for(b=0;b<10;b++)
  {
   for(c=0;c<10;c++)
   {
    for(d=0;d<10;d++)
    {
     for(e=0;e<10;e++)
     {
      for(f=0;f<10;f++)
```

```
    {
     if(a!=0 && d!=0 && a!=b && a!=c && a!=d && a!=e && a!=f && b!=c && b!=d &&
         b!=e && b!=f && c!=d && c!=e && c!=f && d!=e && d!=f && e!=f)
     {
      digits[unique_count]=(a*100) + (b*10) + c;     //an array (digits[]) holding all the
                                                     possible
      unique_count++;                          //combinations of 3 digit numbers where each
                                                     //digit is a unique digit between 1_9 is
                                                     //generated
     }
    }
   }
  }
 }
}

                                  //conditional statements test whether each sum and it's
                                  //corresponding factors meets the conditions of the original
                                  //problem, and the whether duplicates are found is checked
for(a=0;a<10;a++)
{
 for(b=0;b<10;b++)
 {
  for(c=0;c<10;c++)
  {
   for(d=0;d<10;d++)
   {
    for(e=0;e<10;e++)
    {
     for(f=0;f<10;f++)
     {
      if(a!=0 && d!=0 && a!=b && a!=c && a!=d && a!=e && a!=f && b!=c && b!=d &&
          b!=e && b!=f && c!=d && c!=e && c!=f && d!=e && d!=f && e!=f)
      {
       for(j=0;j<1000;j++)
       {
        if(digits[j]!=((d*100)+(e*10)+f))
        {
         ones = c+f;
         if(ones > 9)
         {
          tens = b + e + 1;
          ones = ones%10;
         }
         else
```

```
          {
           tens = b + e;
          }
          if(tens > 9)
          {
           hundreds = a + d + 1;
           tens = tens%10;
          }
          else
          {
           hundreds = a + d;
          }
         }
        }
if (fp == NULL)
 { printf("Error opening file.\n"); }
else{
if(hundreds!=tens && hundreds!=ones && tens!=ones && (hundreds%11!=0)
&& (hundreds%10!=a) && (hundreds%10!=b) && (hundreds%10!=c) &&
(hundreds%10!=d)
&& (hundreds%10!=e) && (hundreds%10!=f) && tens!=a && tens!=b && tens!=c &&
tens!=d
&& tens!=e && tens!=f && ones!=a && ones!=b && ones!=c && ones!=d && ones!=e
&& ones!=f)
        {
         sum_count++;
         fprintf(fp,"\n %d %d %d ",a,b,c);          //the data is printed to a file
         fprintf(fp,"\n+%d %d %d",d,e,f);
         fputs("\n_____________",fp);
         fprintf(fp,"\n %d %d %d\n",hundreds,tens,ones);
        }
       }
      }
     }
    }
   }
  }
 }
}

fclose(fp);
 printf("\nThe number of sums found is %d\n", sum_count);
return 0;
}
```

Appendix B: Data From First C Program

```
 1 0 3
+4 6 9
_____
 5 7 2

 1 0 3
+4 7 9
_____
 5 8 2

 1 0 3
+5 7 9
_____
 6 8 2

 1 0 3
+6 4 9
_____
 7 5 2

 1 0 3
+7 4 9
_____
 8 5 2

 1 0 3
+7 5 9
_____
 8 6 2

 1 0 4
+5 7 9
_____
 6 8 3

 1 0 4
+7 5 9
_____
 8 6 3

 1 0 5
+2 6 9
_____
 3 7 4
```

Appendix C: Second C program

```
//numberpuzzle_v2.c
//Author:  Cherie O'Dell
//Date:  June 27, 2001
//This program generates a list of all possible problems with 3 digit sums, where the factors
//are exactly 3 digits, and every digit in the problem is unique; and where the numbers 1_9
//appear in the problem exactly once.

#include <stdlib.h>
#include <stdio.h>

FILE *fp;

main()
{
 //a, b, & c are the 3 digits of the first factor of each generated problem
 //d, e, & f are the 3 digits of the second factor of each generated problem
 //hundreds, tens, & ones are the 3 digits of each generated sum
 //i & j are counters
 //unique_count records the number of unique 3 digit numbers possible
 //sum_count records the number of possible sums which meet the conditions of the problem
 //digits[] holds all possible unique 3 digit numbers
 //unique_num is a temporary variable holding a unique 3 digit number
 //unique_sum is a temporary variable holding a unique sum

  int a, b, c, d, e, f, i, j, hundreds, tens, ones, digits[2000], unique_num, unique_sum,
sum_count, unique_count;

 unique_num=unique_sum=unique_count=sum_count=0;


 fp = fopen("3dignums.txt", "wt");

 for(i=0;i<2001;i++) digits[i]=0;        //initialize digits[]

 for(a=1;a<10;a++)
 {
  for(b=1;b<10;b++)
  {
   for(c=1;c<10;c++)
   {
```

```
    if(a!=b && a!=c && b!=c)
    {
     unique_num=(a*100) + (b*10) + c;
     digits[unique_count]=unique_num;        //an array (digits[]) holding all the possible
     unique_count++;                         //combinations of 3 digit numbers where each

                                             //digit is a unique digit between 1_9 is generated


   }
  }
 }
}


if (fp == NULL) printf("Error opening file.\n");
else
{
 for(i=0;i<unique_count;i++)
 {
  for(j=0;j<unique_count;j++)
  {
   if(digits[i]!=digits[j])
   {
    unique_sum=digits[i]+digits[j];     //a temporary variable (unique_sum) is set to the sum
    a=digits[i]/100;                    //of each possible combination of unique 3_digit
    a=digits[i]/100;                    //factors which have been generated in digits[]
    b=(digits[i]/10)%10;
    c=digits[i]%10;
    d=digits[j]/100;
    e=(digits[j]/10)%10;
    f=digits[j]%10;
    ones=unique_sum%10;
    tens=(unique_sum/10)%10;
    hundreds=unique_sum/100;

                                //a conditional statement tests whether each sum and it's
                                //corresponding factors meets the conditions of the original
                                //problem
if(a<d && unique_sum<1000 && a!=b && a!=c && a!=d && a!=e && a!=f && a!=ones
&& a!=tens && a!=hundreds && b!=c && b!=d && b!=e && b!=f && b!=ones &&
b!=tens
&& b!=hundreds && c!=d && c!=e && c!=f && c!=ones && c!=tens && c!=hundreds
&& d!=e && d!=f && d!=ones && d!=tens && d!=hundreds && e!=f && e!=ones &&
e!=tens
```

```
&& e!=hundreds&& f!=ones && f!=tens && f!=hundreds&& ones!=tens && ones!=hundreds
&& tens!=hundreds && ones!=0 && tens!=0 && hundreds!=0)
    {
     fprintf(fp,"\t %d %d %d ",a,b,c);   //the data is printed to a file
     fprintf(fp,"\t\t%d %d %d",d,e,f);
     fprintf(fp,"\t\t%d %d %d\n",hundreds,tens,ones);
     sum_count++;
    }
   }
  }
 }
}

fclose(fp);
 printf("\nThe number of sums found is: %d\n", sum_count);
return 0;
}
```

Appendix D: Data From 2nd C Program

659 783 124
739 864 125
359 486 127
368 495 127
367 495 128
439 567 128
357 486 129
438 567 129
654 783 129
735 864 129
658 792 134
729 864 135
429 567 138
654 792 138
428 567 139
725 864 139
596 738 142
695 837 142
586 729 143
692 837 145
583 729 146
592 738 146
487 639 152
784 936 152
629 783 154
638 792 154
782 936 154
329 486 157
482 639 157
634 792 158
327 486 159
624 783 159
387 549 162
783 945 162
782 945 163
328 495 167
382 549 167
327 495 168
286 459 173
295 468 173
293 468 175
283 459 176
367 549 182
394 576 182
457 639 182

493 675 182
754 936 182
763 945 182
276 459 183
492 675 183
546 729 183
762 945 183
392 576 184
752 936 184
273 459 186
543 729 186
362 549 187
452 639 187
384 576 192
483 675 192
546 738 192
645 837 192
275 468 193
482 675 193
382 576 194
273 468 195
642 837 195
542 738 196
569 783 214
659 873 214
478 693 215
748 963 215
378 594 216
738 954 216
349 567 218
376 594 218
439 657 218
475 693 218
736 954 218
745 963 218
348 567 219
438 657 219
564 783 219
654 873 219
657 891 234
746 981 235
718 954 236
745 981 236
654 891 237
419 657 238
716 954 238

418 657 239
596 837 241
576 819 243
675 918 243
673 918 245
718 963 245
736 981 245
573 819 246
591 837 246
735 981 246
319 567 248
715 963 248
318 567 249
397 648 251
619 873 254
637 891 254
391 648 257
634 891 257
614 873 259
519 783 264
514 783 269
593 864 271
683 954 271
546 819 273
591 864 273
645 918 273
681 954 273
418 693 275
643 918 275
318 594 276
543 819 276
316 594 278
415 693 278
394 675 281
673 954 281
671 954 283
391 675 284
357 648 291
384 675 291
546 837 291
573 864 291
571 864 293
381 675 294
541 837 296
351 648 297
658 972 314

529 846 317
628 945 317
627 945 318
654 972 318
527 846 319
567 891 324
657 981 324
519 846 327
564 891 327
618 945 327
654 981 327
617 945 328
517 846 329
586 927 341
576 918 342
572 918 346
581 927 346
467 819 352
618 972 354
627 981 354
462 819 357
624 981 357
614 972 358
457 819 362
527 891 364
452 819 367
524 891 367
546 918 372
542 918 376
546 927 381
541 927 386

Appendix E: Pascal Program Number 1

```
{ This program will list 3-digit numbers of the form a + b = c such that
  all the digits 1 thru 9 are represented among a, b, and c.}

program Sums(input, output, datafile);

uses crt;

type numtype = array[1..25] of string[1];

const size = 9;

var
    a,b,c, count: integer;
    s1,s2,s3,s  : string[10];

datafile:text;

{-------------------------------------------------------------------------}

Function order(s : string):string;
var i,j,index: integer;
         temp: string[10];
            x: numtype;
begin
  index := size;
   for I:= 1 to length (s) do
      x[i]:= copy (s,i,1);

   for i := 1 to index -1 do
    for j := i + 1 to index do
      begin
        if X[i] > X[j] then
          begin
            temp := X[i];
            X[i] := X[j];
            X[j] := temp
          end;
      end;

   s := '';

   for I:= 1 to size do
      s:= s+copy (x[i],1,1);
  Order := s;
end;

{-------------------------------------------------------------------------}

FUNCTION testsum(s:string):BOOLEAN;

var  i: integer;
     x: numtype;

begin
```

```
  s:= order(s);
  if s = '123456789' then testsum := TRUE
                     else testsum := FALSE;
end;


{-------------------------------------------------------------------------}

procedure writefile (var dfile: text; index, x, y, z :integer);
var i : integer;

begin

     writeln(dfile, x:4,y:4,z:4,index:4);

end;


{=========================================================================}
{                              MAIN PROGRAM                               }
{=========================================================================}

begin
  assign(datafile,'sumsdatx.dat');
  rewrite(datafile);

  count :=0;
  for a := 123 to 386 do
    for  b := 123 to 876 do
         begin
           c := a + b;
           if c <= 987 then

           begin
             str(a,s1);  str(b,s2);  str(c,s3);
             s:= s1+s2+s3;
             if (TESTSUM(S)) and (a<b) and (b<c) then

                begin
                  count := count +1;
                  writeln (a:5,b:5 ,c:5,count:12);
                  writefile(datafile,count,a,b,c);
                end;
           end;

         end;
  close(datafile);
end.•
```

Appendix F: BASE 10 DATA Pascal Program Number 1

A	B	C	NO.	A	B	C	NO.	A	B	C	NO.
173	286	459	1	215	478	693	57	234	657	891	113
176	283	459	2	218	475	693	58	237	654	891	114
183	276	459	3	275	418	693	59	254	637	891	115
186	273	459	4	278	415	693	60	257	634	891	116
173	295	468	5	143	586	729	61	324	567	891	117
175	293	468	6	146	583	729	62	327	564	891	118
193	275	468	7	183	546	729	63	364	527	891	119
195	273	468	8	186	543	729	64	367	524	891	120
127	359	486	9	142	596	738	65	243	675	918	121
129	357	486	10	146	592	738	66	245	673	918	122
157	329	486	11	192	546	738	67	273	645	918	123
159	327	486	12	196	542	738	68	275	643	918	124
127	368	495	13	124	659	783	69	342	576	918	125
128	367	495	14	129	654	783	70	346	572	918	126
167	328	495	15	154	629	783	71	372	546	918	127
168	327	495	16	159	624	783	72	376	542	918	128
162	387	549	17	214	569	783	73	341	586	927	129
167	382	549	18	219	564	783	74	346	581	927	130
182	367	549	19	264	519	783	75	381	546	927	131
187	362	549	20	269	514	783	76	386	541	927	132
128	439	567	21	134	658	792	77	152	784	936	133
129	438	567	22	138	654	792	78	154	782	936	134
138	429	567	23	154	638	792	79	182	754	936	135
139	428	567	24	158	634	792	80	184	752	936	136
218	349	567	25	243	576	819	81	162	783	945	137
219	348	567	26	246	573	819	82	163	782	945	138
248	319	567	27	273	546	819	83	182	763	945	139
249	318	567	28	276	543	819	84	183	762	945	140
182	394	576	29	352	467	819	85	317	628	945	141
184	392	576	30	357	462	819	86	318	627	945	142
192	384	576	31	362	457	819	87	327	618	945	143
194	382	576	32	367	452	819	88	328	617	945	144
216	378	594	33	142	695	837	89	216	738	954	145
218	376	594	34	145	692	837	90	218	736	954	146
276	318	594	35	192	645	837	91	236	718	954	147
278	316	594	36	195	642	837	92	238	716	954	148
152	487	639	37	241	596	837	93	271	683	954	149
157	482	639	38	246	591	837	94	273	681	954	150
182	457	639	39	291	546	837	95	281	673	954	151
187	452	639	40	296	541	837	96	283	671	954	152
251	397	648	41	317	529	846	97	215	748	963	153
257	391	648	42	319	527	846	98	218	745	963	154
291	357	648	43	327	519	846	99	245	718	963	155
297	351	648	44	329	517	846	100	248	715	963	156
218	439	657	45	125	739	864	101	314	658	972	157
219	438	657	46	129	735	864	102	318	654	972	158
238	419	657	47	135	729	864	103	354	618	972	159
239	418	657	48	139	725	864	104	358	614	972	160
182	493	675	49	271	593	864	105	235	746	981	161
183	492	675	50	273	591	864	106	236	745	981	162
192	483	675	51	291	573	864	107	245	736	981	163
193	482	675	52	293	571	864	108	246	735	981	164
281	394	675	53	214	659	873	109	324	657	981	165
284	391	675	54	219	654	873	110	327	654	981	166
291	384	675	55	254	619	873	111	354	627	981	167
294	381	675	56	259	614	873	112	357	624	981	168

Appendix G: Pascal Program Number 2

```
    { This program will list base 13 4-digit numbers of the form a + b = c
such that
  all the digits 1 thru F are represented among a, b, and c.}

program Sums(input, output, datafile);

uses crt;

type numtype = array[1..25] of string[1];
     digittype = array[1..25] of longint;
const size = 12;

var
    a,b,c, count: longint;
    s1,s2,s3,s  : string[15];
    s4,s5,s6:string;

datafile:text;
{--------------------------------------------------------------------------}

Function inttostr(i : longint):string;
var s:string[16];
begin
  str(i,s);
  inttostr:=s;
 end;
{--------------------------------------------------------------------------}

Function order(s : string):string;
var i,j,index: integer;
          temp: string[15];
             x: numtype;
begin
  index := size;
   for I:= 1 to length (s) do
      x[i]:= copy (s,i,1);

   for i := 1 to index -1 do
    for j := i + 1 to index do
      begin
        if X[i] > X[j] then
          begin
            temp := X[i];
            X[i] := X[j];
            X[j] := temp
          end;
      end;

   s := '';

   for I:= 1 to size do
      s:= s+copy (x[i],1,1);
  Order := s;
end;
```

```
{------------------------------------------------------------------------}

FUNCTION testsum(s:string):BOOLEAN;

{var  i: integer;
     x: numtype;}

begin

  s:= order(s);
  if s = '123456789ABC' then testsum := TRUE
                           else testsum := FALSE;
end;


{------------------------------------------------------------------------}

procedure writefile (var dfile: text; index, x, y, z :longint);
var i : integer;

begin

     writeln(dfile, x:4,y:4,z:4,index:4);

end;
{------------------------------------------------------------------------}

FUNCTION hex(j:longint):string;
var x,i:integer;
  q,n:longint;
    r:digittype;
    alpha:string[20];
    s:string;

begin
   s:= ''; n:=j;
   x:=0;
   while n>13 do
     begin
       x:=x+1;
       q:=n div 13;
       r[x]:= n mod 13;
       n := q;
     end;
     x:=x+1; r[x]:= n mod 13;
   for i := 1 to x do
     begin
       if r[i]>9
        then
         case r[i] of
           10: alpha:='A';
           11: alpha:='B';
           12: alpha:='C';
         end{case}
        else  alpha := inttostr(r[i]);
              s:=alpha+s;
```

```
     end;
   hex:=s;
end;



{=====================================================================}
{                              MAIN PROGRAM                           }
{=====================================================================}

begin
  clrscr;
  assign(datafile,'decsum1.dat');
  rewrite(datafile);

 count :=0;
  a:= 2578;
  while a <= 20000 do
  begin
    a:=a+1;
    b:= a;
    c:= a+b;
    while c <= 30742 do
    begin
      b:= b+1;
  writeln(a:10,b:10,c:10,hex(a):10,hex(b):10,hex(c):10);
{             if copy(hex(a),1,1) = copy(hex(b),1,1) then
b:=b+13*13*13*13;
             if copy(hex(a),2,1) = copy(hex(b),1,1) then b:=b+13*13*13*13;
             if copy(hex(a),3,1) = copy(hex(b),1,1) then b:=b+13*13*13*13;
             if copy(hex(a),4,1) = copy(hex(b),1,1) then b:=b+13*13*13*13;
             if copy(hex(a),1,1) = copy(hex(b),2,1) then b:=b+13*13*13;
             if copy(hex(a),2,1) = copy(hex(b),2,1) then b:=b+13*13*13;
             if copy(hex(a),3,1) = copy(hex(b),2,1) then b:=b+13*13*13;
             if copy(hex(a),4,1) = copy(hex(b),2,1) then b:=b+13*13*13;
             if copy(hex(a),1,1) = copy(hex(b),3,1) then b:=b+13*13;
             if copy(hex(a),2,1) = copy(hex(b),3,1) then b:=b+13*13;
             if copy(hex(a),3,1) = copy(hex(b),3,1) then b:=b+13*13;
             if copy(hex(a),4,1) = copy(hex(b),3,1) then b:=b+13*13;
             if copy(hex(a),1,1) = copy(hex(b),4,1) then b:=b+13;
             if copy(hex(a),2,1) = copy(hex(b),4,1) then b:=b+13;
             if copy(hex(a),3,1) = copy(hex(b),4,1) then b:=b+13;
             if copy(hex(a),4,1) = copy(hex(b),4,1) then b:=b+13;
             if copy(hex(b),1,1) = copy(hex(b),2,1) then b:=b+13*13*13;
             if copy(hex(b),1,1) = copy(hex(b),3,1) then b:=b+13*13;
             if copy(hex(b),2,1) = copy(hex(b),3,1) then b:=b+13*13;
             if copy(hex(b),1,1) = copy(hex(b),4,1) then b:=b+13;
             if copy(hex(b),2,1) = copy(hex(b),4,1) then b:=b+13;
             if copy(hex(b),3,1) = copy(hex(b),4,1) then b:=b+13;
             if copy(hex(a),1,1) = copy(hex(a),4,1) then a:=a+13;
             if copy(hex(a),1,1) = copy(hex(a),3,1) then a:=a+169;
             if copy(hex(a),1,1) = copy(hex(a),2,1) then a:=a+2197;
             if copy(hex(a),2,1) = copy(hex(a),4,1) then a:=a+13;
             if copy(hex(a),2,1) = copy(hex(a),3,1) then a:=a+169;
             if copy(hex(a),3,1) = copy(hex(a),4,1) then a:=a+13;
             if copy(hex(a),4,1) = '0' then a:=a+13;
             if copy(hex(a),3,1) = '0' then a:=a+169;
```

```
              if copy(hex(a),2,1) = '0' then a:=a+2197;
              if copy(hex(b),4,1) = '0' then b:=b+13;
              if copy(hex(b),3,1) = '0' then b:=b+169;
              if copy(hex(b),2,1) = '0' then b:=b+13;}

            c:= a+b;
            if a + b = c then

            begin
               s:= hex(a)+hex(b)+hex(c);

              if (TESTSUM(S)) and (a<b) and (b<c) then

                 begin
                   count := count +1;
                   writeln(a:10,b:10,c:10,hex(a):10,hex(b):10,hex(c):10);
                   s4:=hex(a); s5:=hex(b); s6:=hex(c);

writeln(datafile,count:10,a:10,b:10,c:10,s4:10,s5:10,s6:10);
                end;
            end;
    end;
  end;
  close(datafile);
end.



{------------------------------------------------------------------------}
•
```

Appendix H: BASE 13 DATA Pascal Program Number 2

No.	A	B	C	A	B	C
1	2579	14980	17559	1235	6A84	7CB9
2	2579	16990	19569	1235	796C	8BA4
3	2579	19324	21903	1235	8A46	9C7B
4	2579	20560	23139	1235	9487	A6BC
5	2579	20890	23469	1235	967C	A8B4
6	2579	25276	27855	1235	B674	C8A9
7	2580	17145	19725	1236	7A5B	8C94
8	2580	19323	21903	1236	8A45	9C7B
9	2580	20733	23313	1236	958B	A7C4
10	2580	23217	25797	1236	A74C	B985
11	2580	25419	27999	1236	B754	C98A
12	2581	12458	15039	1237	5894	6ACB
13	2581	12620	15201	1237	598A	6BC4
14	2581	20558	23139	1237	9485	A6BC
15	2581	20564	23145	1237	948B	A6C5
16	2581	25256	27837	1237	B65A	C894
17	2582	10399	12981	1238	496C	5BA7
18	2582	12601	15183	1238	5974	6BAC
19	2582	12775	15357	1238	5A79	6CB4
20	2582	22723	25305	1238	A45C	B697
21	2582	22897	25479	1238	A564	B79C
22	2582	25099	27681	1238	B569	C7A4
23	2583	12738	15321	1239	5A4B	6C87
24	2583	12774	15357	1239	5A78	6CB4
25	2583	25074	27657	1239	B54A	C786
26	2583	25098	27681	1239	B568	C7A4
27	2584	9905	12489	123A	467C	58B9
28	2584	12617	15201	123A	5987	6BC4
29	2584	25073	27657	123A	B549	C786
30	2584	25253	27837	123A	B657	C894
31	2585	12232	14817	123B	574C	698A
32	2585	12736	15321	123B	5A49	6C87
33	2585	16132	18717	123B	745C	869A
34	2585	17140	19725	123B	7A56	8C94
35	2585	20560	23145	123B	9487	A6C5
36	2585	20728	23313	123B	9586	A7C4
37	2586	9903	12489	123C	467A	58B9
38	2586	10395	12981	123C	4968	5BA7
39	2586	12231	14817	123C	574B	698A
40	2586	16131	18717	123C	745B	869A
41	2586	16983	19569	123C	7965	8BA4
42	2586	20883	23469	123C	9675	A8B4
43	2586	22719	25305	123C	A458	B697

44	2586	23211	25797	123C	A746	B985
45	2590	12437	15027	1243	5879	6ABC
46	2590	12617	15207	1243	5987	6BCA
47	2590	14945	17535	1243	6A58	7C9B
48	2590	22925	25515	1243	A586	B7C9
49	2590	25253	27843	1243	B657	C89A
50	2590	25433	28023	1243	B765	C9A8
51	2592	18921	21513	1245	87C6	9A3B
52	2592	19311	21903	1245	8A36	9C7B
53	2592	20889	23481	1245	967B	A8C3
54	2592	25431	28023	1245	B763	C9A8
55	2593	18914	21507	1246	87BC	9A35
56	2593	18920	21513	1246	87C5	9A3B
57	2593	19310	21903	1246	8A35	9C7B
58	2593	22922	25515	1246	A583	B7C9
59	2593	23204	25797	1246	A73C	B985
60	2594	12613	15207	1247	5983	6BCA
61	2594	20773	23367	1247	95BC	A836
62	2594	25249	27843	1247	B653	C89A
63	2594	25255	27849	1247	B659	C8A3
64	2595	14856	17451	1248	69BA	7C35
65	2595	14940	17535	1248	6A53	7C9B
66	2595	20370	22965	1248	936C	A5B7
67	2595	20382	22977	1248	937B	A5C6
68	2596	12323	14919	1249	57BC	6A38
69	2596	12431	15027	1249	5873	6ABC
70	2596	12725	15321	1249	5A3B	6C87
71	2596	25061	27657	1249	B53A	C786
72	2596	25253	27849	1249	B657	C8A3
73	2597	14182	16779	124A	65BC	7839
74	2597	14854	17451	124A	69B8	7C35
75	2597	15976	18573	124A	736C	85B9
76	2597	25060	27657	124A	B539	C786
77	2598	12219	14817	124B	573C	698A
78	2598	12723	15321	124B	5A39	6C87
79	2598	20379	22977	124B	9378	A5C6
80	2598	20883	23481	124B	9675	A8C3
81	2599	12218	14817	124C	573B	698A
82	2599	12320	14919	124C	57B9	6A38
83	2599	14180	16779	124C	65BA	7839
84	2599	15974	18573	124C	736A	85B9
85	2599	18908	21507	124C	87B6	9A35
86	2599	20366	22965	124C	9368	A5B7
87	2599	20768	23367	124C	95B7	A836
88	2599	23198	25797	124C	A736	B985
89	2603	14932	17535	1253	6A48	7C9B

90	2603	19348	21951	1253	8A64	9CB7
91	2603	20548	23151	1253	9478	A6CB
92	2603	25240	27843	1253	B647	C89A
93	2604	17049	19653	1254	79B6	8C3A
94	2604	19347	21951	1254	8A63	9CB7
95	2604	25395	27999	1254	B736	C98A
96	2606	17041	19647	1256	79AB	8C34
97	2606	17047	19653	1256	79B4	8C3A
98	2606	17053	19659	1256	79BA	8C43
99	2606	17119	19725	1256	7A3B	8C94
100	2606	25393	27999	1256	B734	C98A
101	2607	25230	27837	1257	B63A	C894
102	2607	25236	27843	1257	B643	C89A
103	2607	25242	27849	1257	B649	C8A3
104	2608	14927	17535	1258	6A43	7C9B
105	2608	20435	23043	1258	93BC	A647
106	2608	20543	23151	1258	9473	A6CB
107	2608	22697	25305	1258	A43C	B697
108	2609	8032	10641	1259	386B	4AC7
109	2609	25240	27849	1259	B647	C8A3
110	2610	7863	10473	125A	376B	49C8
111	2610	16041	18651	125A	73BC	8649
112	2610	17049	19659	125A	79B6	8C43
113	2610	25227	27837	125A	B637	C894
114	2611	7862	10473	125B	376A	49C8
115	2611	8030	10641	125B	3869	4AC7
116	2611	16106	18717	125B	743C	869A
117	2611	17036	19647	125B	79A6	8C34
118	2611	17114	19725	125B	7A36	8C94
119	2612	16039	18651	125C	73BA	8649
120	2612	16105	18717	125C	743B	869A
121	2612	20431	23043	125C	93B8	A647
122	2612	22693	25305	125C	A438	B697
123	2616	19335	21951	1263	8A54	9CB7
124	2616	20613	23229	1263	94C8	A75B
125	2616	21285	23901	1263	98C4	AB57
126	2616	25407	28023	1263	B745	C9A8
127	2617	18914	21531	1264	87BC	9A53
128	2617	19334	21951	1264	8A53	9CB7
129	2617	21284	23901	1264	98C3	AB57
130	2617	22862	25479	1264	A538	B79C
131	2618	16951	19569	1265	793C	8BA4
132	2618	17041	19659	1265	79AB	8C43
133	2618	25405	28023	1265	B743	C9A8
134	2620	25139	27759	1267	B59A	C834
135	2620	25151	27771	1267	B5A9	C843

136	2621	10360	12981	1268	493C	5BA7
137	2621	20344	22965	1268	934C	A5B7
138	2621	20608	23229	1268	94C3	A75B
139	2621	22858	25479	1268	A534	B79C
140	2621	24970	27591	1268	B49A	C735
141	2621	25060	27681	1268	B539	C7A4
142	2622	7929	10551	1269	37BC	4A58
143	2622	8019	10641	1269	385B	4AC7
144	2622	25059	27681	1269	B538	C7A4
145	2622	25149	27771	1269	B5A7	C843
146	2623	7850	10473	126A	375B	49C8
147	2623	15950	18573	126A	734C	85B9
148	2623	24968	27591	126A	B498	C735
149	2623	25136	27759	126A	B597	C834
150	2624	7849	10473	126B	375A	49C8
151	2624	8017	10641	126B	3859	4AC7
152	2624	17035	19659	126B	79A5	8C43
153	2625	7926	10551	126C	37B9	4A58
154	2625	10356	12981	126C	4938	5BA7
155	2625	15948	18573	126C	734A	85B9
156	2625	16944	19569	126C	7935	8BA4
157	2625	18906	21531	126C	87B4	9A53
158	2625	20340	22965	126C	9348	A5B7
159	2629	12398	15027	1273	5849	6ABC
160	2629	12476	15105	1273	58A9	6B4C
161	2629	12644	15273	1273	59A8	6C4B
162	2629	20522	23151	1273	9458	A6CB
163	2629	25148	27777	1273	B5A6	C849
164	2629	25316	27945	1273	B6A5	C948
165	2630	12553	15183	1274	5938	6BAC
166	2630	20431	23061	1274	93B8	A65C
167	2630	20773	23403	1274	95BC	A863
168	2630	25135	27765	1274	B596	C83A
169	2630	25225	27855	1274	B635	C8A9
170	2631	20838	23469	1275	963C	A8B4
171	2631	20850	23481	1275	964B	A8C3
172	2631	23100	25731	1275	A68C	B934
173	2631	25224	27855	1275	B634	C8A9
174	2631	25314	27945	1275	B6A3	C948
175	2632	25133	27765	1276	B594	C83A
176	2632	25139	27771	1276	B59A	C843
177	2632	25145	27777	1276	B5A3	C849
178	2634	8253	10887	1278	39AB	4C56
179	2634	8265	10899	1278	39BA	4C65
180	2634	12549	15183	1278	5934	6BAC
181	2634	12639	15273	1278	59A3	6C4B

182	2634	12723	15357	1278	5A39	6CB4
183	2634	20343	22977	1278	934B	A5C6
184	2634	20427	23061	1278	93B4	A65C
185	2634	20517	23151	1278	9453	A6CB
186	2634	24801	27435	1278	B39A	C645
187	2634	24813	27447	1278	B3A9	C654
188	2635	12392	15027	1279	5843	6ABC
189	2635	12470	15105	1279	58A3	6B4C
190	2635	12722	15357	1279	5A38	6CB4
191	2635	24812	27447	1279	B3A8	C654
192	2636	7591	10227	127A	35BC	4869
193	2636	8263	10899	127A	39B8	4C65
194	2636	9853	12489	127A	463C	58B9
195	2636	24799	27435	127A	B398	C645
196	2636	25135	27771	127A	B596	C843
197	2637	8250	10887	127B	39A8	4C56
198	2637	9918	12555	127B	468C	593A
199	2637	20340	22977	127B	9348	A5C6
200	2637	20844	23481	127B	9645	A8C3
201	2638	7589	10227	127C	35BA	4869
202	2638	9851	12489	127C	463A	58B9
203	2638	9917	12555	127C	468B	593A
204	2638	20765	23403	127C	95B4	A863
205	2638	20831	23469	127C	9635	A8B4
206	2638	23093	25731	127C	A685	B934
207	2642	10123	12765	1283	47B9	5A6C
208	2642	10459	13101	1283	49B7	5C6A
209	2642	12565	15207	1283	5947	6BCA
210	2642	22873	25515	1283	A546	B7C9
211	2642	24979	27621	1283	B4A6	C759
212	2642	25315	27957	1283	B6A4	C957
213	2643	14916	17559	1284	6A35	7CB9
214	2643	22638	25281	1284	A3C5	B679
215	2643	25314	27957	1284	B6A3	C957
216	2644	14915	17559	1285	6A34	7CB9
217	2644	20435	23079	1285	93BC	A674
218	2644	20495	23139	1285	9437	A6BC
219	2644	22601	25245	1285	A397	B64C
220	2644	22637	25281	1285	A3C4	B679
221	2644	23087	25731	1285	A67C	B934
222	2645	10462	13107	1286	49BA	5C73
223	2645	20668	23313	1286	953B	A7C4
224	2645	22870	25515	1286	A543	B7C9
225	2645	24970	27615	1286	B49A	C753
226	2645	24976	27621	1286	B4A3	C759
227	2646	8253	10899	1287	39AB	4C65

228	2646	10455	13101	1287	49B3	5C6A
229	2646	12555	15201	1287	593A	6BC4
230	2646	12561	15207	1287	5943	6BCA
231	2646	20493	23139	1287	9435	A6BC
232	2646	20499	23145	1287	943B	A6C5
233	2646	22599	25245	1287	A395	B64C
234	2646	24801	27447	1287	B39A	C654
235	2648	10117	12765	1289	47B3	5A6C
236	2649	9450	12099	128A	43BC	5679
237	2649	10458	13107	128A	49B6	5C73
238	2649	12552	15201	128A	5937	6BC4
239	2649	24798	27447	128A	B397	C654
240	2649	24966	27615	128A	B496	C753
241	2650	8249	10899	128B	39A7	4C65
242	2650	9905	12555	128B	467C	593A
243	2650	20495	23145	128B	9437	A6C5
244	2650	20663	23313	128B	9536	A7C4
245	2651	9448	12099	128C	43BA	5679
246	2651	9904	12555	128C	467B	593A
247	2651	20428	23079	128C	93B5	A674
248	2651	23080	25731	128C	A675	B934
249	2655	24978	27633	1293	B4A5	C768
250	2655	25146	27801	1293	B5A4	C867
251	2656	12323	14979	1294	57BC	6A83
252	2656	12383	15039	1294	5837	6ACB
253	2656	12479	15135	1294	58AC	6B73
254	2656	25109	27765	1294	B576	C83A
255	2656	25145	27801	1294	B5A3	C867
256	2657	7936	10593	1295	37C6	4A8B
257	2657	8080	10737	1295	38A7	4B6C
258	2657	22588	25245	1295	A387	B64C
259	2657	24976	27633	1295	B4A3	C768
260	2658	7929	10587	1296	37BC	4A85
261	2658	7935	10593	1296	37C5	4A8B
262	2658	8085	10743	1296	38AC	4B75
263	2658	24957	27615	1296	B48A	C753
264	2658	25107	27765	1296	B574	C83A
265	2658	25113	27771	1296	B57A	C843
266	2659	8078	10737	1297	38A5	4B6C
267	2659	12380	15039	1297	5834	6ACB
268	2659	22586	25245	1297	A385	B64C
269	2659	24788	27447	1297	B38A	C654
270	2659	25100	27759	1297	B56A	C834
271	2660	24775	27435	1298	B37A	C645
272	2660	24931	27591	1298	B46A	C735
273	2662	24773	27435	129A	B378	C645

274	2662	24785	27447	129A	B387	C654
275	2662	24929	27591	129A	B468	C735
276	2662	24953	27615	129A	B486	C753
277	2662	25097	27759	129A	B567	C834
278	2662	25109	27771	129A	B576	C843
279	2664	7923	10587	129C	37B6	4A85
280	2664	8079	10743	129C	38A6	4B75
281	2664	12315	14979	129C	57B4	6A83
282	2664	12471	15135	129C	58A4	6B73
283	2668	11813	14481	12A3	54B9	678C
284	2668	11825	14493	12A3	54C8	679B
285	2668	12437	15105	12A3	5879	6B4C
286	2668	12497	15165	12A3	58C4	6B97
287	2668	12605	15273	12A3	5978	6C4B
288	2668	12653	15321	12A3	59B4	6C87
289	2668	24953	27621	12A3	B486	C759
290	2668	24965	27633	12A3	B495	C768
291	2668	25109	27777	12A3	B576	C849
292	2668	25133	27801	12A3	B594	C867
293	2668	25277	27945	12A3	B675	C948
294	2668	25289	27957	12A3	B684	C957
295	2669	12466	15135	12A4	589C	6B73
296	2669	12496	15165	12A4	58C3	6B97
297	2669	12652	15321	12A4	59B3	6C87
298	2669	14182	16851	12A4	65BC	7893
299	2669	25132	27801	12A4	B593	C867
300	2669	25288	27957	12A4	B683	C957
301	2670	7755	10425	12A5	36B7	498C
302	2670	8067	10737	12A5	3897	4B6C
303	2670	16041	18711	12A5	73BC	8694
304	2670	16989	19659	12A5	796B	8C43
305	2670	24963	27633	12A5	B493	C768
306	2670	25275	27945	12A5	B673	C948
307	2671	8072	10743	12A6	389C	4B75
308	2671	16976	19647	12A6	795B	8C34
309	2671	24950	27621	12A6	B483	C759
310	2671	25106	27777	12A6	B573	C849
311	2672	7591	10263	12A7	35BC	4896
312	2672	7753	10425	12A7	36B5	498C
313	2672	8065	10737	12A7	3895	4B6C
314	2672	8227	10899	12A7	398B	4C65
315	2672	25099	27771	12A7	B569	C843
316	2673	8214	10887	12A8	397B	4C56
317	2673	9450	12123	12A8	43BC	5697
318	2673	11820	14493	12A8	54C3	679B
319	2673	12600	15273	12A8	5973	6C4B

320	2673	24774	27447	12A8	B379	C654
321	2674	11807	14481	12A9	54B3	678C
322	2674	12431	15105	12A9	5873	6B4C
323	2674	24773	27447	12A9	B378	C654
324	2674	25097	27771	12A9	B567	C843
325	2676	8211	10887	12AB	3978	4C56
326	2676	8223	10899	12AB	3987	4C65
327	2676	16971	19647	12AB	7956	8C34
328	2676	16983	19659	12AB	7965	8C43
329	2677	7586	10263	12AC	35B7	4896
330	2677	8066	10743	12AC	3896	4B75
331	2677	9446	12123	12AC	43B8	5697
332	2677	12458	15135	12AC	5894	6B73
333	2677	14174				

Appendix H: Pascal Program Number 3

```
{ This program will list hexadecimal 5-digit numbers of the form a + b = c
such that
  all the digits 1 thru F are represented among a, b, and c.}

program Sums(input, output, datafile);

uses crt;

type numtype = array[1..25] of string[1];
     digittype = array[1..25] of longint;
const size = 15;

var
    a,b,c, count: longint;
    s1,s2,s3,s  : string[15];
    s4,s5,s6:string;

datafile:text;
{--------------------------------------------------------------------------}

Function inttostr(i : longint):string;
var s:string[16];
begin
  str(i,s);
  inttostr:=s;
 end;
{--------------------------------------------------------------------------}

Function order(s : string):string;
var i,j,index: integer;
         temp: string[15];
            x: numtype;
begin
  index := size;
   for I:= 1 to length (s) do
      x[i]:= copy (s,i,1);

   for i := 1 to index -1 do
    for j := i + 1 to index do
      begin
        if X[i] > X[j] then
          begin
            temp := X[i];
            X[i] := X[j];
            X[j] := temp
          end;
      end;

   s := '';

   for I:= 1 to size do
      s:= s+copy (x[i],1,1);
  Order := s;
end;
```

```
{------------------------------------------------------------------------}

FUNCTION testsum(s:string):BOOLEAN;

{var  i: integer;
     x: numtype;}

begin

  s:= order(s);
  if s = '123456789ABCDEF' then testsum := TRUE
                           else testsum := FALSE;
end;


{------------------------------------------------------------------------}

procedure writefile (var dfile: text; index, x, y, z :longint);
var i : integer;

begin

     writeln(dfile, x:4,y:4,z:4,index:4);

end;
{------------------------------------------------------------------------}

FUNCTION hex(j:longint):string;
var x,i:integer;
  q,n:longint;
    r:digittype;
    alpha:string[20];
    s:string;

begin
   s:= ''; n:=j;
   x:=0;
   while n>16 do
     begin
       x:=x+1;
       q:=n div 16;
       r[x]:= n mod 16;
       n := q;
     end;
     x:=x+1; r[x]:= n mod 16;
   for i := 1 to x do
     begin
       if r[i]>9
        then
         case r[i] of
           10: alpha:='A';
           11: alpha:='B';
           12: alpha:='C';
           13: alpha:='D';
           14: alpha:='E';
           15: alpha:='F';
```

```
         end{case}
        else  alpha := inttostr(r[i]);
              s:=alpha+s;
     end;
   hex:=s;
end;



{=============================================================================}
{                              MAIN PROGRAM
}
{=============================================================================}

begin
  clrscr;
  assign(datafile,'hexsum1.dat');
  rewrite(datafile);

 count :=0;
  a:= 74564;
  while a <= 74579 do
  begin
    a:=a+1;
    b:= a;
    c:= a+b;
    while c <= 1043925 do
    begin
      b:= b+1;

             if copy(hex(a),1,1) = copy(hex(b),1,1) then b:=b+16*16*16*16;
             if copy(hex(a),2,1) = copy(hex(b),1,1) then b:=b+16*16*16*16;
             if copy(hex(a),3,1) = copy(hex(b),1,1) then b:=b+16*16*16*16;
             if copy(hex(a),4,1) = copy(hex(b),1,1) then b:=b+16*16*16*16;
             if copy(hex(a),5,1) = copy(hex(b),1,1) then b:=b+16*16*16*16;
             if copy(hex(a),1,1) = copy(hex(b),2,1) then b:=b+16*16*16;
             if copy(hex(a),2,1) = copy(hex(b),2,1) then b:=b+16*16*16;
             if copy(hex(a),3,1) = copy(hex(b),2,1) then b:=b+16*16*16;
             if copy(hex(a),4,1) = copy(hex(b),2,1) then b:=b+16*16*16;
             if copy(hex(a),5,1) = copy(hex(b),2,1) then b:=b+16*16*16;
             if copy(hex(a),1,1) = copy(hex(b),3,1) then b:=b+16*16;
             if copy(hex(a),2,1) = copy(hex(b),3,1) then b:=b+16*16;
             if copy(hex(a),3,1) = copy(hex(b),3,1) then b:=b+16*16;
             if copy(hex(a),4,1) = copy(hex(b),3,1) then b:=b+16*16;
             if copy(hex(a),5,1) = copy(hex(b),3,1) then b:=b+16*16;
             if copy(hex(a),1,1) = copy(hex(b),4,1) then b:=b+16;
             if copy(hex(a),2,1) = copy(hex(b),4,1) then b:=b+16;
             if copy(hex(a),3,1) = copy(hex(b),4,1) then b:=b+16;
             if copy(hex(a),4,1) = copy(hex(b),4,1) then b:=b+16;
             if copy(hex(a),5,1) = copy(hex(b),4,1) then b:=b+16;
             if copy(hex(b),1,1) = copy(hex(b),2,1) then b:=b+16*16*16;
             if copy(hex(b),1,1) = copy(hex(b),3,1) then b:=b+16*16;
             if copy(hex(b),2,1) = copy(hex(b),3,1) then b:=b+16*16;
             if copy(hex(b),1,1) = copy(hex(b),4,1) then b:=b+16;
             if copy(hex(b),2,1) = copy(hex(b),4,1) then b:=b+16;
             if copy(hex(b),3,1) = copy(hex(b),4,1) then b:=b+16;
             if copy(hex(a),1,1) = copy(hex(a),5,1) then a:=a+1;
```

```
              if copy(hex(a),1,1) = copy(hex(a),4,1) then a:=a+16;
              if copy(hex(a),1,1) = copy(hex(a),3,1) then a:=a+256;
              if copy(hex(a),1,1) = copy(hex(a),2,1) then a:=a+4096;
              if copy(hex(a),2,1) = copy(hex(a),5,1) then a:=a+1;
              if copy(hex(a),2,1) = copy(hex(a),4,1) then a:=a+16;
              if copy(hex(a),2,1) = copy(hex(a),3,1) then a:=a+256;
              if copy(hex(a),3,1) = copy(hex(a),5,1) then a:=a+1;
              if copy(hex(a),3,1) = copy(hex(a),4,1) then a:=a+16;
              if copy(hex(a),4,1) = copy(hex(a),5,1) then a:=a+1;
              if copy(hex(a),5,1) = '0' then a:=a+1;
              if copy(hex(a),4,1) = '0' then a:=a+16;
              if copy(hex(a),3,1) = '0' then a:=a+256;
              if copy(hex(a),2,1) = '0' then a:=a+4096;
              if copy(hex(b),5,1) = '0' then b:=b+1;
              if copy(hex(b),4,1) = '0' then b:=b+16;
              if copy(hex(b),3,1) = '0' then b:=b+256;
              if copy(hex(b),2,1) = '0' then b:=b+4096;

            c:= a+b;
            if a + b = c then

            begin
               s:= hex(a)+hex(b)+hex(c);

              if (TESTSUM(S)) and (a<b) and (b<c) then

                 begin
                   count := count +1;
                   writeln(a:10,b:10,c:10,hex(a):10,hex(b):10,hex(c):10);
                   s4:=hex(a); s5:=hex(b); s6:=hex(c);

writeln(datafile,count:10,a:10,b:10,c:10,s4:10,s5:10,s6:10);
                end;
            end;
    end;
  end;
  close(datafile);
end.



{------------------------------------------------------------------------}
•
```

Appendix J: BASE 16 DATA Pascal Program Number 3

No.	A	B	C	A	B	C
1	74580	433290	507870	12354	69C8A	7BFDE
2	74580	444585	519165	12354	6C8A9	7EBFD
3	74580	507030	581610	12354	7BC96	8DFEA
4	74580	514410	588990	12354	7D96A	8FCBE
5	74580	580470	655050	12354	8DB76	9FECA
6	74580	812955	887535	12354	C679B	D8AEF
7	74580	879015	953595	12354	D69A7	E8CFB
8	74580	886395	960975	12354	D867B	EA9CF
9	74580	948840	1023420	12354	E7A68	F9DBC
10	74580	960135	1034715	12354	EA687	FC9DB
11	74582	507028	581610	12356	7BC94	8DFEA
12	74582	580468	655050	12356	8DB74	9FECA
13	74582	705673	780255	12356	AC489	BE7DF
14	74582	776008	850590	12356	BD748	CFA9E
15	74582	824488	899070	12356	C94A8	DB7FE
16	74582	960583	1035165	12356	EA847	FCB9D
17	74583	637167	711750	12357	9B8EF	ADC46
18	74583	739992	814575	12357	B4A98	C6DEF
19	74583	756207	830790	12357	B89EF	CAD46
20	74583	879012	953595	12357	D69A4	E8CFB
21	74583	960132	1034715	12357	EA684	FC9DB
22	74583	960582	1035165	12357	EA846	FCB9D
23	74584	739991	814575	12358	B4A97	C6DEF
24	74584	776006	850590	12358	BD746	CFA9E
25	74584	824486	899070	12358	C94A6	DB7FE
26	74584	825071	899655	12358	C96EF	DBA47
27	74584	948836	1023420	12358	E7A64	F9DBC
28	74585	444580	519165	12359	6C8A4	7EBFD
29	74585	502765	577350	12359	7ABED	8CF46
30	74585	705670	780255	12359	AC486	BE7DF
31	74586	433284	507870	1235A	69C84	7BFDE
32	74586	440559	515145	1235A	6B8EF	7DC49
33	74586	514404	588990	1235A	7D964	8FCBE
34	74586	579564	654150	1235A	8D7EC	9FB46
35	74586	813039	887625	1235A	C67EF	D8B49
36	74587	579308	653895	1235B	8D6EC	9FA47
37	74587	812948	887535	1235B	C6794	D8AEF
38	74587	886388	960975	1235B	D8674	EA9CF
39	74588	428527	503115	1235C	689EF	7AD4B
40	74588	429037	503625	1235C	68BED	7AF49
41	74588	579307	653895	1235C	8D6EB	9FA47
42	74588	579562	654150	1235C	8D7EA	9FB46
43	74589	429036	503625	1235D	68BEC	7AF49
44	74589	497391	571980	1235D	796EF	8BA4C
45	74589	502761	577350	1235D	7ABE9	8CF46
46	74589	616431	691020	1235D	967EF	A8B4C
47	74591	428524	503115	1235F	689EC	7AD4B
48	74591	440554	515145	1235F	6B8EA	7DC49
49	74591	497389	571980	1235F	796ED	8BA4C
50	74591	616429	691020	1235F	967ED	A8B4C
51	74591	637159	711750	1235F	9B8E7	ADC46
52	74591	756199	830790	1235F	B89E7	CAD46

53	74591	813034	887625	1235F	C67EA	D8B49
54	74591	825064	899655	1235F	C96E8	DBA47
55	74596	514394	588990	12364	7D95A	8FCBE
56	74596	640379	714975	12364	9C57B	AE8DF
57	74596	679049	753645	12364	A5C89	B7FED
58	74596	776069	850665	12364	BD785	CFAE9
59	74596	809114	883710	12364	C589A	D7BFE
60	74596	894359	968955	12364	DA597	EC8FB
61	74596	948824	1023420	12364	E7A58	F9DBC
62	74596	960629	1035225	12364	EA875	FCBD9
63	74597	506953	581550	12365	7BC49	8DFAE
64	74597	580423	655020	12365	8DB47	9FEAC
65	74597	705688	780285	12365	AC498	BE7FD
66	74597	776068	850665	12365	BD784	CFAE9
67	74597	824458	899055	12365	C948A	DB7EF
68	74597	960628	1035225	12365	EA874	FCBD9
69	74599	568286	642885	12367	8ABDE	9CF45
70	74599	568301	642900	12367	8ABED	9CF54
71	74599	580421	655020	12367	8DB45	9FEAC
72	74599	894356	968955	12367	DA594	EC8FB
73	74600	645100	719700	12368	9D7EC	AFB54
74	74600	705685	780285	12368	AC495	BE7FD
75	74600	706015	780615	12368	AC5DF	BE947
76	74600	948820	1023420	12368	E7A54	F9DBC
77	74601	506949	581550	12369	7BC45	8DFAE
78	74601	679044	753645	12369	A5C84	B7FED
79	74601	744159	818760	12369	B5ADF	C7E48
80	74601	952284	1026885	12369	E87DC	FAB45
81	74602	514388	588990	1236A	7D954	8FCBE
82	74602	809108	883710	1236A	C5894	D7BFE
83	74602	824453	899055	1236A	C9485	DB7EF
84	74602	948443	1023045	1236A	E78DB	F9C45
85	74603	509407	584010	1236B	7C5DF	8E94A
86	74603	640372	714975	1236B	9C574	AE8DF
87	74603	644332	718935	1236B	9D4EC	AF857
88	74603	816367	890970	1236B	C74EF	D985A
89	74603	948442	1023045	1236B	E78DA	F9C45
90	74603	951772	1026375	1236B	E85DC	FA947
91	74604	547551	622155	1236C	85ADF	97E4B
92	74604	547806	622410	1236C	85BDE	97F4A
93	74604	644331	718935	1236C	9D4EB	AF857
94	74604	645096	719700	1236C	9D7E8	AFB54
95	74604	951771	1026375	1236C	E85DB	FA947
96	74604	952281	1026885	1236C	E87D9	FAB45
97	74605	568295	642900	1236D	8ABE7	9CF54
98	74605	685295	759900	1236D	A74EF	B985C
99	74606	547804	622410	1236E	85BDC	97F4A
100	74606	568279	642885	1236E	8ABD7	9CF45
101	74607	509403	584010	1236F	7C5DB	8E94A
102	74607	547548	622155	1236F	85ADC	97E4B
103	74607	685293	759900	1236F	A74ED	B985C
104	74607	706008	780615	1236F	AC5D8	BE947
105	74607	744153	818760	1236F	B5AD9	C7E48
106	74607	816363	890970	1236F	C74EB	D985A
107	74612	371593	446205	12374	5AB89	6CEFD
108	74612	375178	449790	12374	5B98A	6DCFE
109	74612	580438	655050	12374	8DB56	9FECA

110	74612	640363	714975	12374	9C56B	AE8DF
111	74612	886363	960975	12374	D865B	EA9CF
112	74612	960613	1035225	12374	EA865	FCBD9
113	74613	637167	711780	12375	9B8EF	ADC64
114	74613	739977	814590	12375	B4A89	C6DFE
115	74613	756207	830820	12375	B89EF	CAD64
116	74613	878922	953535	12375	D694A	E8CBF
117	74613	960072	1034685	12375	EA648	FC9BD
118	74613	960612	1035225	12375	EA864	FCBD9
119	74614	568286	642900	12376	8ABDE	9CF54
120	74614	572111	646725	12376	8BACF	9DE45
121	74614	580436	655050	12376	8DB54	9FECA
122	74614	894281	968895	12376	DA549	EC8BF
123	74616	710094	784710	12378	AD5CE	BF946
124	74616	710124	784740	12378	AD5EC	BF964
125	74616	956109	1030725	12378	E96CD	FBA45
126	74616	956124	1030740	12378	E96DC	FBA54
127	74616	960069	1034685	12378	EA645	FC9BD
128	74617	371588	446205	12379	5AB84	6CEFD
129	74617	375503	450120	12379	5BACF	6DE48
130	74617	709868	784485	12379	AD4EC	BF865
131	74617	739973	814590	12379	B4A85	C6DFE
132	74617	740063	814680	12379	B4ADF	C6E58
133	74617	894278	968895	12379	DA546	EC8BF
134	74618	309487	384105	1237A	4B8EF	5DC69
135	74618	375172	449790	1237A	5B984	6DCFE
136	74618	878917	953535	1237A	D6945	E8CBF
137	74618	944587	1019205	1237A	E69CB	F8D45
138	74618	955612	1030230	1237A	E94DC	FB856
139	74619	640356	714975	1237B	9C564	AE8DF
140	74619	878031	952650	1237B	D65CF	E894A
141	74619	886356	960975	1237B	D8654	EA9CF
142	74619	944586	1019205	1237B	E69CA	F8D45
143	74620	297455	372075	1237C	489EF	5AD6B
144	74620	297965	372585	1237C	48BED	5AF69
145	74620	543455	618075	1237C	84ADF	96E5B
146	74620	543710	618330	1237C	84BDE	96F5A
147	74620	709865	784485	1237C	AD4E9	BF865
148	74620	710120	784740	1237C	AD5E8	BF964
149	74620	955610	1030230	1237C	E94DA	FB856
150	74620	956120	1030740	1237C	E96D8	FBA54
151	74621	297964	372585	1237D	48BEC	5AF69
152	74621	956104	1030725	1237D	E96C8	FBA45
153	74622	543708	618330	1237E	84BDC	96F5A
154	74622	568278	642900	1237E	8ABD6	9CF54
155	74622	681423	756045	1237E	A65CF	B894D
156	74622	710088	784710	1237E	AD5C8	BF946
157	74623	297452	372075	1237F	489EC	5AD6B
158	74623	309482	384105	1237F	4B8EA	5DC69
159	74623	375497	450120	1237F	5BAC9	6DE48
160	74623	543452	618075	1237F	84ADC	96E5B
161	74623	572102	646725	1237F	8BAC6	9DE45
162	74623	637157	711780	1237F	9B8E5	ADC64
163	74623	681422	756045	1237F	A65CE	B894D
164	74623	740057	814680	1237F	B4AD9	C6E58
165	74623	756197	830820	1237F	B89E5	CAD64
166	74623	878027	952650	1237F	D65CB	E894A

167	74628	371577	446205	12384	5AB79	6CEFD
168	74628	375162	449790	12384	5B97A	6DCFE
169	74628	433242	507870	12384	69C5A	7BFDE
170	74628	679017	753645	12384	A5C69	B7FED
171	74628	776037	850665	12384	BD765	CFAE9
172	74628	960087	1034715	12384	EA657	FC9DB
173	74629	640991	715620	12385	9C7DF	AEB64
174	74629	739961	814590	12385	B4A79	C6DFE
175	74629	752351	826980	12385	B7ADF	C9E64
176	74629	776036	850665	12385	BD764	CFAE9
177	74629	824426	899055	12385	C946A	DB7EF
178	74629	825071	899700	12385	C96EF	DBA74
179	74629	948806	1023435	12385	E7A46	F9DCB
180	74630	645070	719700	12386	9D7CE	AFB54
181	74630	705625	780255	12386	AC459	BE7DF
182	74630	817855	892485	12386	C7ABF	D9E45
183	74630	948805	1023435	12386	E7A45	F9DCB
184	74631	894399	969030	12387	DA5BF	EC946
185	74631	956109	1030740	12387	E96CD	FBA54
186	74631	960084	1034715	12387	EA654	FC9DB
187	74633	306142	380775	12389	4ABDE	5CF67
188	74633	306157	380790	12389	4ABED	5CF76
189	74633	371572	446205	12389	5AB74	6CEFD
190	74633	449212	523845	12389	6DABC	7FE45
191	74633	449227	523860	12389	6DACB	7FE54
192	74633	679012	753645	12389	A5C64	B7FED
193	74633	705622	780255	12389	AC456	BE7DF
194	74633	739957	814590	12389	B4A75	C6DFE
195	74634	313311	387945	1238A	4C7DF	5EB69
196	74634	375156	449790	1238A	5B974	6DCFE
197	74634	433236	507870	1238A	69C54	7BFDE
198	74634	824421	899055	1238A	C9465	DB7EF
199	74634	870351	944985	1238A	D47CF	E6B59
200	74634	940476	1015110	1238A	E59BC	F7D46
201	74635	449225	523860	1238B	6DAC9	7FE54
202	74635	804335	878970	1238B	C45EF	D697A
203	74636	293599	368235	1238C	47ADF	59E6B
204	74636	293854	368490	1238C	47BDE	59F6A
205	74636	447934	522570	1238C	6D5BE	7F94A
206	74636	449209	523845	1238C	6DAB9	7FE45
207	74636	939709	1014345	1238C	E56BD	F7A49
208	74636	940474	1015110	1238C	E59BA	F7D46
209	74637	300783	375420	1238D	496EF	5BA7C
210	74637	306153	380790	1238D	4ABE9	5CF76
211	74637	359103	433740	1238D	57ABF	69E4C
212	74637	673263	747900	1238D	A45EF	B697C
213	74637	939708	1014345	1238D	E56BC	F7A49
214	74637	956103	1030740	1238D	E96C7	FBA54
215	74638	293852	368490	1238E	47BDC	59F6A
216	74638	306137	380775	1238E	4ABD9	5CF67
217	74638	435647	510285	1238E	6A5BF	7C94D
218	74638	447932	522570	1238E	6D5BC	7F94A
219	74638	608207	682845	1238E	947CF	A6B5D
220	74638	645062	719700	1238E	9D7C6	AFB54
221	74639	293596	368235	1238F	47ADC	59E6B
222	74639	300781	375420	1238F	496ED	5BA7C
223	74639	313306	387945	1238F	4C7DA	5EB69

224	74639	359101	433740	1238F	57ABD	69E4C
225	74639	435646	510285	1238F	6A5BE	7C94D
226	74639	608206	682845	1238F	947CE	A6B5D
227	74639	640981	715620	1238F	9C7D5	AEB64
228	74639	673261	747900	1238F	A45ED	B697C
229	74639	752341	826980	1238F	B7AD5	C9E64
230	74639	804331	878970	1238F	C45EB	D697A
231	74639	817846	892485	1238F	C7AB6	D9E45
232	74639	825061	899700	1238F	C96E5	DBA74
233	74639	870346	944985	1238F	D47CA	E6B59
234	74639	894391	969030	1238F	DA5B7	EC946
235	74644	506966	581610	12394	7BC56	8DFEA
236	74644	809066	883710	12394	C586A	D7BFE
237	74644	812891	887535	12394	C675B	D8AEF
238	74644	894311	968955	12394	DA567	EC8FB
239	74645	444490	519135	12395	6C84A	7EBDF
240	74645	506575	581220	12395	7BACF	8DE64
241	74645	705640	780285	12395	AC468	BE7FD
242	74645	748255	822900	12395	B6ADF	C8E74
243	74645	886735	961380	12395	D87CF	EAB64
244	74646	506964	581610	12396	7BC54	8DFEA
245	74646	678984	753630	12396	A5C48	B7FDE
246	74646	705759	780405	12396	AC4DF	BE875
247	74647	371528	446175	12397	5AB48	6CEDF
248	74647	371693	446340	12397	5ABED	6CF84
249	74647	709838	784485	12397	AD4CE	BF865
250	74647	739928	814575	12397	B4A58	C6DEF
251	74647	813998	888645	12397	C6BAE	D8F45
252	74647	894143	968790	12397	DA4BF	EC856
253	74647	894308	968955	12397	DA564	EC8FB
254	74648	306142	380790	12398	4ABDE	5CF76
255	74648	309967	384615	12398	4BACF	5DE67
256	74648	371527	446175	12398	5AB47	6CEDF
257	74648	445357	520005	12398	6CBAD	7EF45
258	74648	449212	523860	12398	6DABC	7FE54
259	74648	678982	753630	12398	A5C46	B7FDE
260	74648	705637	780285	12398	AC465	BE7FD
261	74648	739927	814575	12398	B4A57	C6DEF
262	74648	805567	880215	12398	C4ABF	D6E57
263	74648	809902	884550	12398	C5BAE	D7F46
264	74650	317390	392040	1239A	4D7CE	5FB68
265	74650	317420	392070	1239A	4D7EC	5FB86
266	74650	444485	519135	1239A	6C845	7EBDF
267	74650	809060	883710	1239A	C5864	D7BFE
268	74650	936125	1010775	1239A	E48BD	F6C57
269	74650	936155	1010805	1239A	E48DB	F6C75
270	74651	317164	391815	1239B	4D6EC	5FA87
271	74651	378079	452730	1239B	5C4DF	6E87A
272	74651	812884	887535	1239B	C6754	D8AEF
273	74651	873679	948330	1239B	D54CF	E786A
274	74651	936154	1010805	1239B	E48DA	F6C75
275	74652	289503	364155	1239C	46ADF	58E7B
276	74652	289758	364410	1239C	46BDE	58F7A
277	74652	317163	391815	1239C	4D6EB	5FA87
278	74652	317418	392070	1239C	4D7EA	5FB86
279	74652	447678	522330	1239C	6D4BE	7F85A
280	74652	449208	523860	1239C	6DAB8	7FE54

281	74653	371687	446340	1239D	5ABE7	6CF84
282	74653	445352	520005	1239D	6CBA8	7EF45
283	74653	477887	552540	1239D	74ABF	86E5C
284	74653	936122	1010775	1239D	E48BA	F6C57
285	74654	289756	364410	1239E	46BDC	58F7A
286	74654	296911	371565	1239E	487CF	5AB6D
287	74654	306136	380790	1239E	4ABD8	5CF76
288	74654	317386	392040	1239E	4D7CA	5FB68
289	74654	435391	510045	1239E	6A4BF	7C85D
290	74654	447676	522330	1239E	6D4BC	7F85A
291	74654	677071	751725	1239E	A54CF	B786D
292	74654	709831	784485	1239E	AD4C7	BF865
293	74654	809896	884550	1239E	C5BA8	D7F46
294	74654	813991	888645	1239E	C6BA7	D8F45
295	74655	289500	364155	1239F	46ADC	58E7B
296	74655	296910	371565	1239F	487CE	5AB6D
297	74655	309960	384615	1239F	4BAC8	5DE67
298	74655	378075	452730	1239F	5C4DB	6E87A
299	74655	435390	510045	1239F	6A4BE	7C85D
300	74655	477885	552540	1239F	74ABD	86E5C
301	74655	506565	581220	1239F	7BAC5	8DE64
302	74655	677070	751725	1239F	A54CE	B786D
303	74655	705750	780405	1239F	AC4D6	BE875
304	74655	748245	822900	1239F	B6AD5	C8E74
305	74655	805560	880215	1239F	C4AB8	D6E57
306	74655	873675	948330	1239F	D54CB	E786A
307	74655	886725	961380	1239F	D87C5	EAB64
308	74655	894135	968790	1239F	DA4B7	EC856
309	74660	444505	519165	123A4	6C859	7EBFD
310	74660	878935	953595	123A4	D6957	E8CFB
311	74661	433224	507885	123A5	69C48	7BFED
312	74661	440559	515220	123A5	6B8EF	7DC94
313	74661	510399	585060	123A5	7C9BF	8ED64
314	74661	514374	589035	123A5	7D946	8FCEB
315	74661	813039	887700	123A5	C67EF	D8B94
316	74661	882879	957540	123A5	D78BF	E9C64
317	74662	514373	589035	123A6	7D945	8FCEB
318	74662	809033	883695	123A6	C5849	D7BEF
319	74662	824408	899070	123A6	C9458	DB7FE
320	74662	890063	964725	123A6	D94CF	EB875
321	74663	309487	384150	123A7	4B8EF	5DC96
322	74663	375112	449775	123A7	5B948	6DCEF
323	74663	813982	888645	123A7	C6B9E	D8F45
324	74663	878932	953595	123A7	D6954	E8CFB
325	74664	313791	388455	123A8	4C9BF	5ED67
326	74664	375111	449775	123A8	5B947	6DCEF
327	74664	382956	457620	123A8	5D7EC	6FB94
328	74664	433221	507885	123A8	69C45	7BFED
329	74664	445341	520005	123A8	6CB9D	7EF45
330	74664	809886	884550	123A8	C5B9E	D7F46
331	74664	824406	899070	123A8	C9456	DB7FE
332	74664	940476	1015140	123A8	E59BC	F7D64
333	74665	317630	392295	123A9	4D8BE	5FC67
334	74665	444500	519165	123A9	6C854	7EBFD
335	74665	809030	883695	123A9	C5846	D7BEF
336	74665	935900	1010565	123A9	E47DC	F6B85
337	74667	382158	456825	123AB	5D4CE	6F879

338	74667	382188	456855	123AB	5D4EC	6F897
339	74667	935373	1010040	123AB	E45CD	F6978
340	74667	935388	1010055	123AB	E45DC	F6987
341	74668	382187	456855	123AC	5D4EB	6F897
342	74668	382952	457620	123AC	5D7E8	6FB94
343	74668	935387	1010055	123AC	E45DB	F6987
344	74668	935897	1010565	123AC	E47D9	F6B85
345	74668	939197	1013865	123AC	E54BD	F7869
346	74668	940472	1015140	123AC	E59B8	F7D64
347	74669	288751	363420	123AD	467EF	58B9C
348	74669	445336	520005	123AD	6CB98	7EF45
349	74669	935371	1010040	123AD	E45CB	F6978
350	74669	939196	1013865	123AD	E54BC	F7869
351	74670	293055	367725	123AE	478BF	59C6D
352	74670	317625	392295	123AE	4D8B9	5FC67
353	74670	365775	440445	123AE	594CF	6B87D
354	74670	382155	456825	123AE	5D4CB	6F879
355	74670	809880	884550	123AE	C5B98	D7F46
356	74670	813975	888645	123AE	C6B97	D8F45
357	74671	288749	363420	123AF	467ED	58B9C
358	74671	293054	367725	123AF	478BE	59C6D
359	74671	309479	384150	123AF	4B8E7	5DC96
360	74671	313784	388455	123AF	4C9B8	5ED67
361	74671	365774	440445	123AF	594CE	6B87D
362	74671	440549	515220	123AF	6B8E5	7DC94
363	74671	510389	585060	123AF	7C9B5	8ED64
364	74671	813029	887700	123AF	C67E5	D8B94
365	74671	882869	957540	123AF	D78B5	E9C64
366	74671	890054	964725	123AF	D94C6	EB875
367	74677	509663	584340	123B5	7C6DF	8EA94
368	74677	510383	585060	123B5	7C9AF	8ED64
369	74677	812873	887550	123B5	C6749	D8AFE
370	74677	882383	957060	123B5	D76CF	E9A84
371	74677	882863	957540	123B5	D78AF	E9C64
372	74677	886343	961020	123B5	D8647	EA9FC
373	74678	640327	715005	123B6	9C547	AE8FD
374	74678	816367	891045	123B6	C74EF	D98A5
375	74678	817807	892485	123B6	C7A8F	D9E45
376	74679	640326	715005	123B7	9C546	AE8FD
377	74679	886341	961020	123B7	D8645	EA9FC
378	74679	894111	968790	123B7	DA49F	EC856
379	74679	894351	969030	123B7	DA58F	EC946
380	74680	313055	387735	123B8	4C6DF	5EA97
381	74680	313775	388455	123B8	4C9AF	5ED67
382	74680	447980	522660	123B8	6D5EC	7F9A4
383	74680	449180	523860	123B8	6DA9C	7FE54
384	74680	804335	879015	123B8	C45EF	D69A7
385	74680	805535	880215	123B8	C4A9F	D6E57
386	74680	939740	1014420	123B8	E56DC	F7A94
387	74680	940460	1015140	123B8	E59AC	F7D64
388	74681	317134	391815	123B9	4D6CE	5FA87
389	74681	317614	392295	123B9	4D8AE	5FC67
390	74681	447724	522405	123B9	6D4EC	7F8A5
391	74681	449164	523845	123B9	6DA8C	7FE45
392	74681	812869	887550	123B9	C6745	D8AFE
393	74682	935373	1010055	123BA	E45CD	F6987
394	74682	936093	1010775	123BA	E489D	F6C57

395	74682	939228	1013910	123BA	E54DC	F7896
396	74682	940428	1015110	123BA	E598C	F7D46
397	74684	447646	522330	123BC	6D49E	7F85A
398	74684	447721	522405	123BC	6D4E9	7F8A5
399	74684	447886	522570	123BC	6D58E	7F94A
400	74684	447976	522660	123BC	6D5E8	7F9A4
401	74684	449161	523845	123BC	6DA89	7FE45
402	74684	449176	523860	123BC	6DA98	7FE54
403	74684	939181	1013865	123BC	E54AD	F7869
404	74684	939226	1013910	123BC	E54DA	F7896
405	74684	939661	1014345	123BC	E568D	F7A49
406	74684	939736	1014420	123BC	E56D8	F7A94
407	74684	940426	1015110	123BC	E598A	F7D46
408	74684	940456	1015140	123BC	E59A8	F7D64
409	74685	357615	432300	123BD	574EF	698AC
410	74685	359055	433740	123BD	57A8F	69E4C
411	74685	476655	551340	123BD	745EF	869AC
412	74685	477855	552540	123BD	74A9F	86E5C
413	74685	935370	1010055	123BD	E45CA	F6987
414	74685	936090	1010775	123BD	E489A	F6C57
415	74685	939180	1013865	123BD	E54AC	F7869
416	74685	939660	1014345	123BD	E568C	F7A49
417	74686	292559	367245	123BE	476CF	59A8D
418	74686	293039	367725	123BE	478AF	59C6D
419	74686	317129	391815	123BE	4D6C9	5FA87
420	74686	317609	392295	123BE	4D8A9	5FC67
421	74686	435359	510045	123BE	6A49F	7C85D
422	74686	435599	510285	123BE	6A58F	7C94D
423	74686	447644	522330	123BE	6D49C	7F85A
424	74686	447884	522570	123BE	6D58C	7F94A
425	74687	292558	367245	123BF	476CE	59A8D
426	74687	293038	367725	123BF	478AE	59C6D
427	74687	313048	387735	123BF	4C6D8	5EA97
428	74687	313768	388455	123BF	4C9A8	5ED67
429	74687	357613	432300	123BF	574ED	698AC
430	74687	359053	433740	123BF	57A8D	69E4C
431	74687	435358	510045	123BF	6A49E	7C85D
432	74687	435598	510285	123BF	6A58E	7C94D
433	74687	476653	551340	123BF	745ED	869AC
434	74687	477853	552540	123BF	74A9D	86E5C
435	74687	509653	584340	123BF	7C6D5	8EA94
436	74687	510373	585060	123BF	7C9A5	8ED64
437	74687	804328	879015	123BF	C45E8	D69A7
438	74687	805528	880215	123BF	C4A98	D6E57
439	74687	816358	891045	123BF	C74E6	D98A5
440	74687	817798	892485	123BF	C7A86	D9E45
441	74687	882373	957060	123BF	D76C5	E9A84
442	74687	882853	957540	123BF	D78A5	E9C64
443	74687	894103	968790	123BF	DA497	EC856
444	74687	894343	969030	123BF	DA587	EC946
445	74693	428527	503220	123C5	689EF	7ADB4
446	74693	506527	581220	123C5	7BA9F	8DE64
447	74693	882367	957060	123C5	D76BF	E9A84
448	74693	886687	961380	123C5	D879F	EAB64
449	74694	547806	622500	123C6	85BDE	97FA4
450	74694	572031	646725	123C6	8BA7F	9DE45
451	74694	645006	719700	123C6	9D78E	AFB54

452	74694	890031	964725	123C6	D94AF	EB875
453	74695	297455	372150	123C7	489EF	5ADB6
454	74695	543710	618405	123C7	84BDE	96FA5
455	74695	709790	784485	123C7	AD49E	BF865
456	74695	956045	1030740	123C7	E968D	FBA54
457	74696	293854	368550	123C8	47BDE	59FA6
458	74696	309919	384615	123C8	4BA9F	5DE67
459	74696	710014	784710	123C8	AD57E	BF946
460	74696	956029	1030725	123C8	E967D	FBA45
461	74697	289758	364455	123C9	46BDE	58FA7
462	74697	317118	391815	123C9	4D6BE	5FA87
463	74697	375423	450120	123C9	5BA7F	6DE48
464	74697	449163	523860	123C9	6DA8B	7FE54
465	74698	317342	392040	123CA	4D79E	5FB68
466	74698	870287	944985	123CA	D478F	E6B59
467	74698	935357	1010055	123CA	E45BD	F6987
468	74698	944507	1019205	123CA	E697B	F8D45
469	74699	382126	456825	123CB	5D4AE	6F879
470	74699	449161	523860	123CB	6DA89	7FE54
471	74699	873631	948330	123CB	D549F	E786A
472	74699	877951	952650	123CB	D657F	E894A
473	74699	935341	1010040	123CB	E45AD	F6978
474	74699	944506	1019205	123CB	E697A	F8D45
475	74701	935339	1010040	123CD	E45AB	F6978
476	74701	935354	1010055	123CD	E45BA	F6987
477	74701	956024	1030725	123CD	E9678	FBA45
478	74701	956039	1030740	123CD	E9687	FBA54