Steps taken to make working program run properly on input file generated by local TBlastX

- Bring up working program (i.e. Parse-Blast-revised.tru)
- Change file path so it points to input file generated by local BlastN
- Run program... "Wrong line in Evaluate score, line 1" encountered
- Turn on echo of input (uncomment line in InputBlast)
- Last line read by program is "Score = 117 bits...". Is the format of the Score line different in CMR Blast compared to local Blast?
- Compare input files generated by the two programs. Indeed, there is a difference in the lines related to score:

  CMR Blast:

```
Score = 131 (25.7 bits), Expect = 2.7e-12, Sum P(3) = 2.7e-12
 Identities = 67/109 (61%), Positives = 67/109 (61%), Strand = Minus / Plus
```

  Local Blast:

```
Score = 46.0 bits (94), Expect = 1e-006
 Identities = 25/48 (52%), Positives = 28/48 (58%)
 Frame = +3 / +2
```

- It's now understandable why the program crashed reading the score. The program line that failed was an error-checking line:

  IF Size(field$) < 8 THEN CAUSE ERROR...

  In the CMR Blast-generated file, the number of fields is certainly greater than the number of fields in the local Blast-generated file. How many fields are there?
- Insert line just before faulty line to read: MAT PRINT field$ and rerun program. This statement prints out all values held within the array (or MATrix) field$
- The program crashes again (of course, since I didn't fix anything), and now the last line is six values, representing the six words of the last line of input.
- Change the offending program line to read:

  IF Size(field$) < 6 THEN CAUSE ERROR

- The only thing I'm getting from this line is the E-value. Check to make sure that the E-value is in field it ought to be in. It's in the sixth field. The next line assigns the sixth field to E-value, so I'm OK.
- Rerun program. Now error occurs in line:

  IF Size(field$) < 9 THEN CAUSE ERROR 2...

- I now know how to handle this. Moving the MAT PRINT field$ line, I find that the total number of fields is really 3, not 9. Make the appropriate change and check whether the value I'm extracting from this line (ID) is in the right field. It is.
- Rerun program. "Subscript out of bounds" Error occurs on the same input line but a different program line:

IF field$(9) = "Plus" THEN...

- I'm now sensitized to numbers of fields and see immediately that there are not 9 fields in the input line (Identities = ...). In fact, there's no "Plus" at all on the line. I check the input file and instead of Plus/Minus strand information, TBlastX provides something to do with "frame", and it's on the NEXT line. I need to read another line before checking for the strand.
- I steal from the same subroutine the following lines, which evidently reads the input file, and insert it into the program before the line that checks for strand:

```
CALL InputBlast
CALL Explode(line$,field$," ()=%")
```

- Rather than figure out how to interpret the frames of the query and the target, I'll just save both and worry about it later. So I insert the two statements:

```
LET query_orientation$ = field$(2)
LET target_orientation$ = field$(4)
```

- Rerun program. Seems to be running without error, so I stop it (Click File in upper left corner, then Stop). Turn off input echoing (by putting a ! before PRINT line$ in *InputBlast*).
- Rerun program. Goes to completion... no it doesn't. It crashes after processing many queries, up until query Contig389-L. Check input file. That contig is the last one in the file. Looks like the program doesn't recognize the logical end of the file.
- Scan program for anything related to end of file. Found it in *Find_seq*. It says that end of file is recognized by the word "Parameters". Indeed, there is the word "Parameters" at the end of the file generated by CMR-Blast. But there is no such word at the end of the locally generated file.
- Look for word comes uniquely at end of file. Change:

```
ELSE IF line$[1:10] = "Parameters" THEN    ! Ran into end of file
```

to

```
ELSE IF line$[1:6] = "Matrix" THEN   ! Ran into end of file
```

- Rerun program. Goes to completion. Output looks OK,... except column 7 is always " ". That can't be right.
- Check routine in program that prints output (*Output_hit*; the only routine that has statements that print to a file). The seventh item output is *orientation$*.
- Where is *orientation$* determined? I type *locate orientation$*. It gives me three lines... oh yes! I obliterated the line that assigned a value to *orientation$* and instead assigned values to *query_orientation$* and *target_orientation$*. Now I need to print out these new variables instead of the nonexistent *orientation$.*
- In *Output_hit*, I replace the line:

```
PRINT #output: orientation$; comma$;
```

With

```
PRINT #output: query_orientation$; comma$; target_orientation$; comma$;
```

- Rerun program. Goes to completion. Output looks OK.