Test File 1

```
HTEST00400000002B
T0040000B544553542053545249 4E47
T00401615000000000000B04401650C00054C00B2C401938401C
M00401D04+TEST
M00402004+TEST
M00402304+TEST
M00402604+TEST
M00402904+TEST
E00401C
```

Test File 2

```
HCOPY000000000052
T0000001E5448495320495320412044C4F4E4745522054455354205354524 94E472054
T00001E144841542049532041205445535420535452494E47
T00003D15000000000000B04003D5080005480322C0040380043
M00004404+COPY
M00004704+COPY
M00004A04+COPY
M00004D04+COPY
M00005004+COPY
E
```

Some Reference Definitions

H-Record

Column	Value
1	H
2-7	Program or Library Name
8-13	Starting Address of object program (hex)
14-19	Length of the object program in byes (hex)

T-Record

Colum	Value
1	T
2-7	Starting Address of object code (hex)
8-9	Length of Object code in bytes (hex)
10-69	Object Code

M-Record

Colum	Value
1	M
2-7	Starting Address of Modification
8-9	Length of the modification (hex)
10	+/-
11-16	Symbol

E-Record

Column	Value
1	E
2-7	Address of First Instruction to be run (hex)

The idea of address normalization/nullification is this.

Length of both programs is being saved.
We obtain a load_point, which is relocation address/loading address.

For given example load_point location is 0x1000 for simplicity of examples.

For each program:
Get length and loading address
Get all records

<u>Getting length and loading address</u>

This information is stored in **H-Record** columns 8-19

Test File 1:

H TEST 004000 00002B

004000 – is the starting address of object program
00002B – is the length of the program in bytes

Test File 2:

H COPY 000000 000052

000000 – is the starting address of the object program
000052 – is the length of the program in bytes

Having this information we subtract starting address from all of the record addresses in the object file

Test File 1:

Subtracting **004000** from all addresses

(before subtraction, address separated for ease of reading)
HTEST **004000** 00002B
T **004000** 0B5445535420535452494E47
T **004016** 1500000000000B04401650C00054C00B2C401938401C
M **00401D** 04+TEST
M **004020** 04+TEST

```
M 004023  04+TEST
M 004026  04+TEST
M 004029 04+TEST
E  00401C
```

(after subtraction)

```
 HTEST 000000 00002B
T  000000 0B5445535420535452494E47
T  000016 1500000000000B04401650C00054C00B2C401938401C
M 00001D 04+TEST
M 000020  04+TEST
M 000023  04+TEST
M 000026  04+TEST
M 000029 04+TEST
E 00001C
```

Test File 2:

Subtracting **000000** from all addresses

(before subtraction, address separated for ease of reading)

```
HCOPY 000000 000052
T  000000 1E5448495320495320412041204C4F4E474552205445535420535452494E472054
T  00001E 1448415420495320412054455354205354524494E47
T  00003D 1500000000000B04003D5080005480322C0040380043
M 000044 04+COPY
M 000047 04+COPY
M 00004A 04+COPY
M 00004D 04+COPY
M 000050  04+COPY
E
```

(after subtraction)

```
HCOPY 000000 000052
T  000000 1E5448495320495320412041204C4F4E474552205445535420535452494E472054
T  00001E 1448415420495320412054455354205354524494E47
T  00003D 1500000000000B04003D5080005480322C0040380043
M 000044 04+COPY
M 000047 04+COPY
M 00004A 04+COPY
M 00004D 04+COPY
M 000050  04+COPY
E
```

What did we just achieve? Now any of these programs can be loaded at any memory location, and all it will take is to just add the load_point to any of the programs. But now we need to combine these programs into one long one.

Now this is a normalization issue.

This is how we can tackle this issue:

We take the lengths of the programs which were given to use, we also make sure we keep the order the same, so Test File 1 is loaded before Test File 2.

Loading Test File 1:

HTEST **000000** 00002B
T **000000** 0B5445535420535452494E47
T **000016** 1500000000000B04401650C00054C00B2C401938401C
M **00001D** 04+TEST
M **000020** 04+TEST
M **000023** 04+TEST
M **000026** 04+TEST
M **000029** 04+TEST
E **00001C**

Appending Test File 2:

HTEST **000000** 00002B
T **000000** 0B5445535420535452494E47
T **000016** 1500000000000B04401650C00054C00B2C401938401C
M **00001D** 04+TEST
M **000020** 04+TEST
M **000023** 04+TEST
M **000026** 04+TEST
M **000029** 04+TEST
E **00001C**
HCOPY **000000** 000052
T **000000** 1E544849532049532041204C4F4E474552205445535420535452494E472054
T **00001E** 144841542049532041205445535420535452494E47
T **00003D** 1500000000000B04003D5080005480322C0040380043
M **000044** 04+COPY
M **000047** 04+COPY
M **00004A** 04+COPY
M **00004D** 04+COPY
M **000050** 04+COPY
E

Now, this does not look right and will crash, we need to do some fixing.
Clean up the E-Records, and H-Records.

Since we nullified the initial memory loading, both of these addresses are loaded into memory **000000**, this means we can now just use one H-Record (as well as one program name), and one E-Record. However, before this is done, we need to account for Test File 1 length and Test File 2 length.
To do this we first add Test File 1 length to all records in Test File 2.

(After adding Test File 1 length to Test File 2 length)

HTEST **000000** 00002B
T **000000** 0B5445535420535452494E47
T **000016** 1500000000000B04401650C00054C00B2C401938401C
M **00001D** 04+TEST
M **000020** 04+TEST
M **000023** 04+TEST
M **000026** 04+TEST
M **000029** 04+TEST
E **00001C**
HCOPY **00002B** 000052
T **00002B** 1E544849532049532041204C4F4E47455220544553542053545452494E472054
T **000049** 14484154204953204120544553542053545452494E47
T **000068** 1500000000000B04003D5080005480322C0040380043
M **00006F** 04+COPY
M **000072** 04+COPY
M **000075** 04+COPY
M **000078** 04+COPY
M **00007B** 04+COPY
E

(To check if the records are correct, you can either add starting address of Test File 2 to length of Test File 2, or add the starting address of last T-Record and add its length, in both cases the final address should be **00007D**)

Now, we need to clean up Test File 1's length, and to do this, we add length of Test File 2 to length of Test File 1.

(After adding lengths together)

HTEST **000000** 00007D
T **000000** 0B5445535420535452494E47
T **000016** 1500000000000B04401650C00054C00B2C401938401C
M **00001D** 04+TEST
M **000020** 04+TEST
M **000023** 04+TEST
M **000026** 04+TEST
M **000029** 04+TEST
E **00001C**
HCOPY **00002B** 000052
T **00002B** 1E544849532049532041204C4F4E47455220544553542053545452494E472054
T **000049** 14484154204953204120544553542053545452494E47
T **000068** 1500000000000B04003D5080005480322C0040380043
M **00006F** 04+COPY
M **000072** 04+COPY
M **000075** 04+COPY
M **000078** 04+COPY
M **00007B** 04+COPY
E

Now we can clean up the E-Records and H-Records, we can also replace M-Record symbol of Test File 2 with the program name of Test File 1.

(After clean up and record rearrangement)

```
HTEST 000000 00007D
T  000000 0B5445535420535452494E47
T  000016 1500000000000B04401650C00054C00B2C401938401C
T  00002B 1E544849532049532041204C4F4E474552205445535420535452494E472054
T  000049 144841542049532041205445535420535452494E47
T  000068 1500000000000B04003D5080005480322C0040380043
M 00001D 04+TEST
M 000020  04+TEST
M 000023  04+TEST
M 000026  04+TEST
M 000029 04+TEST
M 00006F 04+TEST
M 000072 04+TEST
M 000075 04+TEST
M 000078 04+TEST
M 00007B  04+TEST
E  00001C
```

Now everything is in sync, the first instruction of Test File 1 will run and if all is coded properly in SIC it will successfully just reference/jump to Test File 2 code.

Now we can change the initial load_point to our new one.

In our case we have load_point of 0x1000.

To do this we add 0x1000 to all of the addresses (including H-Record and E-Record) referenced by all off the records.

(After adding load_point)

```
HTEST 001000 00007D
T  001000 0B5445535420535452494E47
T  001016 1500000000000B04401650C00054C00B2C401938401C
T  00102B 1E544849532049532041204C4F4E474552205445535420535452494E472054
T  001049 144841542049532041205445535420535452494E47
T  001068 1500000000000B04003D5080005480322C0040380043
M 00101D 04+TEST
M 001020  04+TEST
M 001023  04+TEST
M 001026  04+TEST
M 001029 04+TEST
M 00106F 04+TEST
```

```
M 001072 04+TEST
M 001075 04+TEST
M 001078 04+TEST
M 00107B  04+TEST
E  00101C
```

Now this is our final program.

If more than one link is done, same steps apply to them in a loop, so if we have 3 linking programs, same logic is applied to them and their information is being taken into the account. (Gladly this will not be an issue for T-Records and M-Records due to the way they are stored)

This report does not take into the account D-Records and R-Records, and we will need to discuss what to do with them and how they can impact this solution.