Juan Tarrat

LAB 2 REPORT

1. THEORY TOPICS

In this lab we have learnt about the different data type that exist (bool, char, int, short int, long int, unsigned int, unsigned long int, double, float…) in C and their sizes and how that affects their maximum and minimum values. This values have different sizes depending on the data they are meant to store. This is relevant because memory is a very important part in computer engineering and we need to understand how it works, specially in msp 430.

Endianess refers to the way data is stored in memory. Msp 430 uses what is called a "little endian policy". This means that the least significant byte of a word is stored in the lower address.

2. PROGRAM 1

- **Calculation of data types by hand:**

  Variable: short int (2 bytes) signed *2 bytes = 16 bits*
  MAX_VALUE = $2^{15} - 1 = 32767$ **MIN_VALUE** = $-2^{15} = -32768$

  Variable: unsigned short int (2 bytes) *2 bytes = 16 bits*
  MAX_VALUE = $(2 * 2^{15}) - 1 = 65535$ **MIN_VALUE** = 0

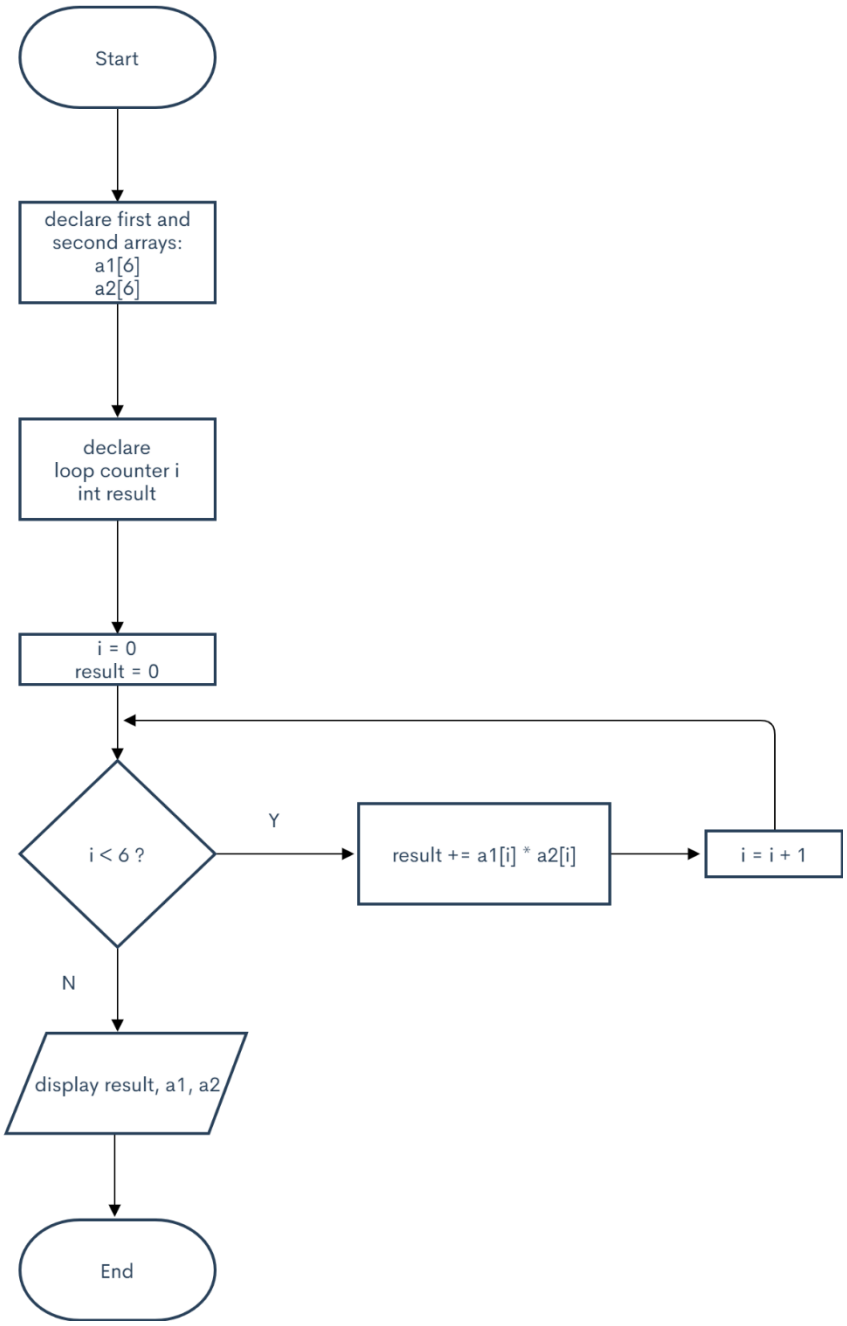
- **Console output:**

```
--------------------------------------------------------------------------------
|Data Type              | Bytes   | Minimum              | Maximum              |
--------------------------------------------------------------------------------
|char                   | 1       |                 -128 | 127                  |
|short int              | 2       |               -32768 | 32767                |
|int                    | 2       |               -32768 | 32767                |
|long int               | 4       |          -2147483648 | 2147483647           |
|long long int          | 8       | -9223372036854775808 | 9223372036854775807  |
|unsigned char          | 1       |                    0 | 255                  |
|unsigned short int     | 2       |                    0 | 65535                |
|unsigned int           | 2       |                    0 | 65535                |
|unsigned long int      | 4       |                    0 | 4294967295           |
|unsigned long long int | 8       |                    0 | 18446744073709551615 |
|float                  | 4       |        1.175494e-38 | 3.402823e+38         |
|double                 | 8       |       2.225074e-308 | 1.797693e+308        |
--------------------------------------------------------------------------------
```

As you can see, the values calculated match those from the table that have been calculated using built in functions within the limits. h and float.h classes.

3. PROGRAM 2

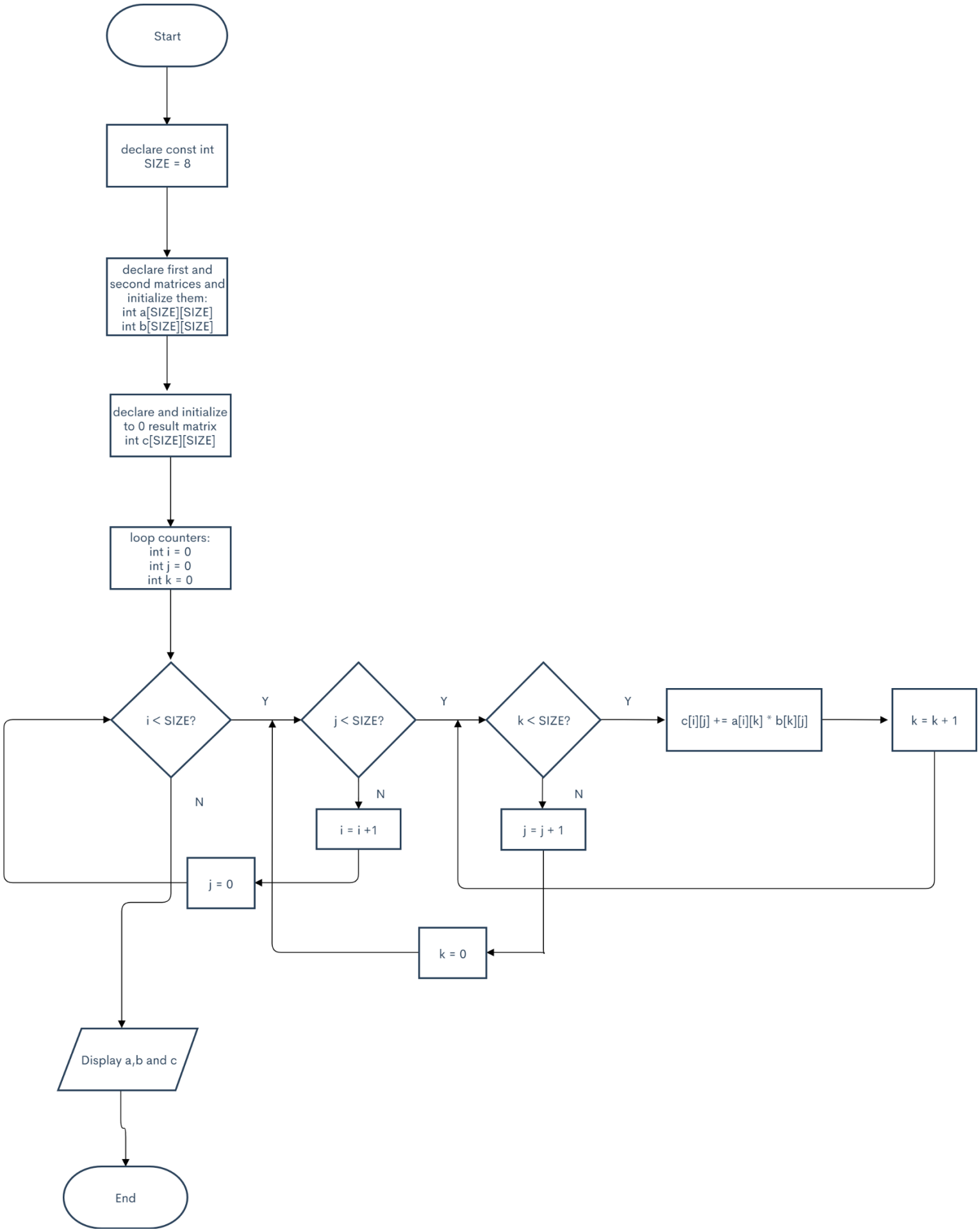
```
Input array a1: [-1 2 5 3 -5 6 ]
Input array a2: [-7 8 23 13 23 28 ]
Dot Product is: 230
```



4. BONUS PROGRAM

```
a =

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

b =

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

a X b =

    36    67    95   112   145   103    70   112
   180   151   174   188   218   199   123   110
   243   210   235   238   269   249   172   168
   181   186   225   225   253   244   175   217
   131   136   181   180   199   147   148   122
   149   154   192   190   212   151   141   124
   137   126   120   142   188   125    75    97
   131   143   215   224   249   191   174   161
```

Juan Tarrat


Start
declare const int SIZE = 8
declare first and second matrices and initialize them: int a[SIZE][SIZE] int b[SIZE][SIZE]
declare and initialize to 0 result matrix int c[SIZE][SIZE]
loop counters: int i = 0 int j = 0 int k = 0
i < SIZE?
Y
j < SIZE?
Y
k < SIZE?
Y
c[i][j] += a[i][k] * b[k][j]
k = k + 1
N
i = i +1
N
j = j + 1
N
j = 0
k = 0
Display a,b and c
End