Yeditepe University
Department of Computer Engineering

CSE 232
Systems Programming
Spring 2019

Term Project

Preprocessor

Due to: 22nd May 2019

3 Students in a Group

In this project, you will develop a **preprocessor** named MTRX that replaces matrix-based macros by the appropriate C codes.

Write your Preprocessor in C and use gcc compiler on Linux.

MTRX preprocessor expands a code as follows:

MTRX macros	**Expanded code (expanded.c)**
Declarations	**Replace the macro with**
@def N 100	#define N 100
@mat A N, M	declaration of a 2-dimensional integer array of size NxM
@vec X N	declaration of a 1-dimensional integer array of size N
@int a	declaration of an integer variable a
Matrix/vector operations	
@A=B+C or @A=B-C	addition/subtraction of two matrices (A,B,C are 2-dimensional arrays)
@A=B*C	Multiplication of two matrices (A,B,C are 2-dimensional arrays)
@X=Y+Z or @ X=Y-Z	addition/subtraction of two vectors (X,Y,Z are 1-dimensional arrays)
@X=Y*Z	Multiplication of two vectors (X,Y,Z are 1-dimensional arrays)
@X=A*Z	matrix-vector multiplication (A is 2-dimensional and X,Y are 1-dimensional arrays)
@X=A*Z +Y	matrix-vector multiplication and addition (A is 2-dimensional and X,Y,Z are 1-dimensional arrays)
@A=a*B	Multiplication of a matrix by a constant (A,B are 2-dimensional arrays)
@X=a*Y	Multiplication of a vector by a constant (X,Y are 1-dimensional arrays)
I/O operations	
@read A < filename	Reads a matrix from a file
@read X < filename	Reads a vector from a file
@print A	Displays the matrix on the screen
@print X	Displays the vector on the screen

Your preprocessor must perform the following:

```
while (not EOF)
        read a line from the source code
        if the line starts with '@'
                parse the line and display the tokens
                if it is a declaration
                        expand the macro with the appropriate C code and write it to the file named expanded.c
                        enter the symbol to symbol table
                otherwise
                        search symbol table for the symbol names
                        expand the macro with the appropriate C code and write it to the file named expanded.c
        otherwise
                write the line to the file expanded.c
```

Use the following structure for symbol table:

```
struct symbol_table {
      int  type;        // 0-integer variable 1-vector 2-matrix
      char name[10];    // array/variable name
      char dim1;        // size of dimension1 (may also be a char array)
      char dim2;        // size of dimension2 (may also be a char array)
}
struct symbol_table ST[20];   // maximum 20 arrays/variables
```

Write a function named searchST() that searches the symbol table for the name of the symbol and returns its index in the table:

```
int searchST(char* name);      //name is the array/variable name
```

Ex:

Source code with MTRX macros	Expanded code (expanded.c)
`@def  N  100`	`#define  N  100`
`@def  M  50`	`#define  M  50`
`@vec  X  M`	`int X[M];`
`@vec  Y  M`	`int Y[M];`
`@mat  A  N,M`	`int A[N][M];`
`int main(){`	`int main(){`

`  @read X < file1`	`FILE *f1;`
	`f1=fopen("file1", "r");`
	`for (int i=0; i<M; i++)`
	`  fscanf(f1, "%c", &X[i]);`

`  @read A < file2`	`FILE *f2;`
	`f2=fopen("file2", "r");`
	`for (int i=0; i<N; i++)`
	`  for (int j=0; j<M; j++)`
	`  fscanf(f2, "%c", &A[i][j]);`

`  @Y=A*X`	`for (int i=0; i<N; i++)`
	`  for (int j=0; j<M; j++)`
	`  Y[j]=A[i][j]*X[j];`

`  @print Y`	`for (int i=0; i<M; i++)` `  printf("%d\n", Y[i]);`
`}`	`}`