Homework 3

In this assignment, you will apply the concepts you learned in Chapters 5 and 6 to the problem of optimizing code for a memory-intensive application. Consider a procedure to copy and transpose the elements of an N x N matrix of type int. That is, for source matrix S and destination matrix D, we want to copy each element s(i,j) to d1(j,i). This code can be written with a simple loop.

```
void transpose (int *dst, int *src, int dim) {
    int i, j;

    for (i = 0; i < dim; i++)
        for (j = 0; j < dim; j++)
            dst[j*dim + i] = src[i*dim + j];
}
```

where the arguments to the procedure are pointers to the destination (dst) and source (src) matrices, as well as the matrix size N (dim).

Your job is to devise a transpose route that runs as fast as possible.

```
5x5 matrix


ABCDE
FGHIJ
KLMNO
PQRST
UVWXY

convert to
vvvvv

AFKPU
BGLQV
CHMRW
DINSX
EJOTY
```

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

int main(int argc, char* argv[])
{
    int dim = 2000;

    int *src = malloc(dim*dim * sizeof(int));
    int *dest = malloc(dim*dim * sizeof(int));

    int  count = 0;
    for(int i = 0; i<dim; i++)
        for(int j = 0; j<dim; j++)
            src[i*dim + j] = count++;


    //time this

    clock_t start, end;
    double cpu_time_used;

    start = clock();

    transpose_old(dest, src, dim);
    //  transpose_new(dest, src, dim);

    end = clock();

    cpu_time_used = ((double)(end-start)) / CLOCKS_PER_SEC;

    printf("%f",cpu_time_used);

    return 0;
}

void transpose (int *dst, int *src, int dim) {
    int B = 16;
    int i, j, k, i1, j1;
    for (i = 0; i < dim; i += B)
        for (j = 0; j < dim; j += B)
            for (i1 = i; (i1 < i+B) && (i1 < dim); i1++) {
                k = i1*dim;
                for (j1 = j; (j1 < j+B) && (j1 < dim); j1++)
                    dst[k + j1] = src[j1*dim + i1];
            }
}
```