```
// matrix multiply routine
void multiply_d1(int arrSize, double **aMatrix, double **bMatrix, double **product)
{
  for(int i=0;i<arrSize;i++) {
    for(int j=0;j<arrSize;j++) {
      double sum = 0;
      for(int k=0;k<arrSize;k++) {
        sum += aMatrix[i][k] * bMatrix[k][j];
      }
      product[i][j] = sum;
    }
  }
}


void multiply_d2(int arrSize, double **aMatrix, double **bMatrix, double **product)
{

        double sum=0;

        for(int i=0; i < arrSize; i++){
                for(int j=0 ; j< arrSize; j++) product[i][j]=0;

                for(int k=0; k < arrSize; k++){
                        double alpha = aMatrix[i][k];
                        for(int j=0; j < arrSize; j++){
                                product[i][j] += alpha*bMatrix[k][j];
                        }
                }
        }


}
```

```
void multiply_d3(int arrSize, double **aMatrix, double **bMatrix, double **product)
{
	/* blocking */
	int blockI = 200;
	int blockJ = 200;
	int blockK = 100;
	double *miniB = (double *)malloc(blockK*blockJ*sizeof(double));
	double *miniA = (double *)malloc(blockI*blockK*sizeof(double));
	double *miniproduct = (double *)malloc(blockI*blockJ*sizeof(double));

	double dot=0.0;
	double aa=0.0;
	for (int i=0; i<arrSize; i+=blockI)
		for(int j=0; j<arrSize; j+=blockJ){

			for(int ii=0;ii<blockI;ii++)
				for(int jj=0;jj<blockJ;jj++)
					miniproduct[ii*blockJ+jj] = 0.0;

			for(int k=0; k<arrSize; k+=blockK){

				/* pre-initializing block B */
				for(int kk=0;kk<blockK;kk++)
					for(int jj=0;jj<blockJ;jj++)
						miniB[kk*blockJ+jj]=bMatrix[k+kk][j+jj];

				/* pre-initializing block A */
				for(int ii=0;ii<blockI;ii++)
					for(int kk=0;kk<blockK;kk++)
						miniA[ii*blockK+kk]=aMatrix[i+ii][k+kk];


				for(int ii=0;ii<blockI;ii++)
					for(int kk=0;kk<blockK;kk++){
						aa=miniA[ii*blockK+kk];
						for(int jj=0;jj<blockJ;jj++)
							miniproduct[ii*blockJ+jj]+=aa*miniB[kk*blockJ+jj];
					}
			}


			for(int ii=0;ii<blockI;ii++)
				for(int jj=0;jj<blockJ;jj++)
					product[i+ii][j+jj] = miniproduct[ii*blockJ+jj];
		}

	free(miniA);
	free(miniB);
	free(miniproduct);
}
```

```
void multiply_d4(int arrSize, double **aMatrix, double **bMatrix, double **product)
{
	/* blocking */
	int blockI = 200;
	int blockJ = 200;
	int blockK = 100;


	#pragma omp parallel
	{
	double *miniB = (double *)malloc(blockK*blockJ*sizeof(double));
	double *miniA = (double *)malloc(blockI*blockK*sizeof(double));
	double *miniproduct = (double *)malloc(blockI*blockJ*sizeof(double));
	double dot=0.0;
	double aa=0.0;


	#pragma omp for collapse(2)
	for (int i=0; i<arrSize; i+=blockI)
		for(int j=0; j<arrSize; j+=blockJ){

			for(int ii=0;ii<blockI;ii++)
				for(int jj=0;jj<blockJ;jj++)
					miniproduct[ii*blockJ+jj] = 0.0;

			for(int k=0; k<arrSize; k+=blockK){

				/* pre-initializing block B */
				for(int kk=0;kk<blockK;kk++)
					for(int jj=0;jj<blockJ;jj++)
						miniB[kk*blockJ+jj]=bMatrix[k+kk][j+jj];

				/* pre-initializing block A */
				for(int ii=0;ii<blockI;ii++)
					for(int kk=0;kk<blockK;kk++)
						miniA[ii*blockK+kk]=aMatrix[i+ii][k+kk];


				for(int ii=0;ii<blockI;ii++)
					for(int kk=0;kk<blockK;kk++){
						aa=miniA[ii*blockK+kk];
						for(int jj=0;jj<blockJ;jj++)
							miniproduct[ii*blockJ+jj]+=aa*miniB[kk*blockJ+jj];
					}
			}


			for(int ii=0;ii<blockI;ii++)
				for(int jj=0;jj<blockJ;jj++)
					product[i+ii][j+jj] = miniproduct[ii*blockJ+jj];
		}


	free(miniA);
	free(miniB);
	free(miniproduct);
	}

}
```

```
void multiply_d5(int arrSize, double **aMatrix, double **bMatrix, double **product,
 struct timeval *startTime,struct timeval  *endTime)
{
	double dot;
	double miniaa;
	/* blocking */
	int blockI = 200;
	int blockJ = 200;
	int blockK = 100;

	#pragma omp parallel
	{
	double *miniA = (double *)malloc(blockI*blockK*sizeof(double));
	double *miniB = (double *)malloc(blockK*blockJ*sizeof(double));
	double *miniproduct = (double *)malloc(blockI*blockJ*sizeof(double));
	double dot;
	int i,j;

//	gettimeofday(startTime, NULL);

	#pragma omp for collapse(2)
	for (i=0; i<arrSize; i+=blockI)
		for(j=0; j<arrSize; j+=blockJ){

			for(int ii=0;ii<blockI;ii++)
				for(int jj=0;jj<blockJ;jj++)
					miniproduct[ii*blockJ+jj] = 0.0;

			for(int k=0; k<arrSize; k+=blockK){

				/* pre-initializing block B */
				for(int kk=0;kk<blockK;kk++)
					for(int jj=0;jj<blockJ;jj++)
						miniB[kk*blockJ+jj]=bMatrix[k+kk][j+jj];

				/* pre-initializing block A */
				for(int ii=0;ii<blockI;ii++)
					for(int kk=0;kk<blockK;kk++)
						miniA[ii*blockK+kk]=aMatrix[i+ii][k+kk];

				cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans,
								blockI, blockJ, blockK,
								1,
								miniA, blockK,
								miniB, blockJ,
								1,
								miniproduct, blockJ);
			}

			for(int ii=0;ii<blockI;ii++)
				for(int jj=0;jj<blockJ;jj++)
					product[i+ii][j+jj] = miniproduct[ii*blockJ+jj];
		}
//	gettimeofday(endTime, NULL);

	free(miniA);
	free(miniB);
	free(miniproduct);

	}


}
```

```
void multiply_d6(int arrSize, double **aMatrix, double **bMatrix, double **product,
 struct timeval *startTime,struct timeval  *endTime)
{

	double *A = (double *)malloc(arrSize*arrSize*sizeof(double));
	double *B = (double *)malloc(arrSize*arrSize*sizeof(double));
	double *C = (double *)malloc(arrSize*arrSize*sizeof(double));

	for(int i=0;i<arrSize;i++)
		for(int j=0;j<arrSize;j++){
			A[i*arrSize+j] = aMatrix[i][j];
			B[i*arrSize+j] = bMatrix[i][j];
		}

	gettimeofday(startTime, NULL);

	cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans,
						arrSize, arrSize, arrSize,
						1,
						A, arrSize,
						B, arrSize,
						0,
						C, arrSize);

	gettimeofday(endTime, NULL);

	for(int i=0;i<arrSize;i++)
		for(int j=0;j<arrSize;j++)
			product[i][j] = C[i*arrSize+j];


	free(A);
	free(B);
	free(C);

}
```