```c
/*
  Basic matrix-matrix multiplication for HPSC OpenMP Assignment
  Michael Oberg, modified from code provided by Ian Karlin

  HPSC
  Aaron Holt
  Assignment 9
*/

#include <stdio.h>
#include <sys/time.h>
#include <sys/types.h>
#include <stdlib.h>

char *program = "OpenMP-manual";

double calctime(struct timeval start, struct timeval end)
{
  double time = 0.0;

  //struct timeval {
  //   time_t      tv_sec;     /* seconds */
  //   suseconds_t tv_usec;    /* microseconds */
  //};
  time = end.tv_usec - start.tv_usec;
  time = time/1000000;
  time += end.tv_sec - start.tv_sec;

  return time;
}

int main()
{
  int    i, j, k;
  int    n = 1024;
  double *A;
  double *B;
  double *C;

  double time;
  struct timeval start;
  struct timeval end;

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

  // Initialize arrays
  for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
    {
       A[i*n + j] = 0.0;
       B[i*n + j] = i + j*n;
       C[i*n + j] = i*n + j;
    }

  int num_threads;
  num_threads = 8;

  // while(num_threads < 66){
              // timed loop
        gettimeofday(&start, NULL);
        /* Insert OpenMP #pragma(s) here */
        omp_set_dynamic(0);     // Explicitly disable dynamic teams
        omp_set_num_threads(num_threads); // Set number of threads for parallel reg
ions to 8
        #pragma omp parallel shared(A, B, C, num_threads) private(i, j, k)
        #pragma omp for nowait
        for (i = 0; i <n; i++)
          for(j = 0; j < n; j++)
            for(k = 0; k < n; k++)
              A[i*n + j] += B[i*n + k] * C[j*n + k];

        num_threads = omp_get_max_threads();
        gettimeofday(&end, NULL);

        // calc & print results
        time = calctime(start, end);
        printf("%s multiplcation time: %lf(s) for size: %dx%d\n", program, time, n,
 n);
        printf("Num threads = %i\n", num_threads);
        // num_threads = num_threads*2;
  // }


  return 0;
}
```

```c
/*
  Basic matrix-matrix multiplication for HPSC OpenMP Assignment
  Michael Oberg, modified from code provided by Ian Karlin
*/

#include <stdio.h>
#include <sys/time.h>
#include <sys/types.h>
#include <stdlib.h>

char *program = "OpenMP-compiletime";

double calctime(struct timeval start, struct timeval end)
{
  double time = 0.0;

  //struct timeval {
  //   time_t      tv_sec;     /* seconds */
  //   suseconds_t tv_usec;    /* microseconds */
  //};
  time = end.tv_usec - start.tv_usec;
  time = time/1000000;
  time += end.tv_sec - start.tv_sec;

  return time;
}

int main()
{
  int    i, j, k;
  int    n = 1024;
  double *A;
  double *B;
  double *C;

  double time;
  struct timeval start;
  struct timeval end;

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

  // Initialize arrays
  for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
    {
       A[i*n + j] = 0.0;
       B[i*n + j] = i + j*n;
       C[i*n + j] = i*n + j;
    }

  int num_threads;
  // num_threads = 1;

  // while(num_threads < 66){
                // timed loop
        gettimeofday(&start, NULL);
        /* Insert OpenMP #pragma(s) here */
        // omp_set_dynamic(0);     // Explicitly disable dynamic teams
        // omp_set_num_threads(num_threads); // Set number of threads for parallel
regions to 8
        #pragma omp parallel shared(A, B, C, num_threads) private(i, j, k)
        #pragma omp for schedule(static) nowait
        for (i = 0; i <n; i++)
          for(j = 0; j < n; j++)
            for(k = 0; k < n; k++)
              A[i*n + j] += B[i*n + k] * C[j*n + k];

        num_threads = omp_get_max_threads();
        gettimeofday(&end, NULL);

        // calc & print results
        time = calctime(start, end);
        printf("%s multiplcation time: %lf(s) for size: %dx%d\n", program, time, n,
 n);
        printf("Num threads = %i\n", num_threads);
        // num_threads = num_threads*2;
  // }

  return 0;
}
```

```
/*
  Basic matrix-matrix multiplication for HPSC OpenMP Assignment
  Michael Oberg, modified from code provided by Ian Karlin
*/

#include <stdio.h>
#include <sys/time.h>
#include <sys/types.h>
#include <stdlib.h>

char *program = "OpenMP-runtime";

double calctime(struct timeval start, struct timeval end)
{
  double time = 0.0;

  //struct timeval {
  //   time_t      tv_sec;     /* seconds */
  //   suseconds_t tv_usec;    /* microseconds */
  //};
  time = end.tv_usec - start.tv_usec;
  time = time/1000000;
  time += end.tv_sec - start.tv_sec;

  return time;
}

int main()
{
  int    i, j, k;
  int    n = 1024;
  double *A;
  double *B;
  double *C;

  double time;
  struct timeval start;
  struct timeval end;

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

  // Initialize arrays
  for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
    {
       A[i*n + j] = 0.0;
       B[i*n + j] = i + j*n;
       C[i*n + j] = i*n + j;
    }

  int num_threads;
  // num_threads = 1;

  // while(num_threads < 66){
                // timed loop
        gettimeofday(&start, NULL);
        /* Insert OpenMP #pragma(s) here */
        // omp_set_num_threads(num_threads); // Set number of threads for parallel
regions to 8
        #pragma omp parallel shared(A, B, C, num_threads) private(i, j, k)
        #pragma omp for schedule(dynamic) nowait
        for (i = 0; i <n; i++)
          for(j = 0; j < n; j++)
            for(k = 0; k < n; k++)
              A[i*n + j] += B[i*n + k] * C[j*n + k];

        num_threads = omp_get_max_threads();
        gettimeofday(&end, NULL);

        // calc & print results
        time = calctime(start, end);
        printf("%s multiplcation time: %lf(s) for size: %dx%d\n", program, time, n,
 n);
        printf("Num threads = %i\n", num_threads);
        // num_threads = num_threads*2;
  // }

  return 0;
}
```