```
/* HEADERS */

#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <math.h>

/*Declare integer, real, and Complex variables */

/* C is case sensitive Nmax is different from nmax
                  and xx   is different from Xx    */

int Nmax ;
double xx;

Nmax = 128    /* not 128. */
  xx = 128.0  /* not 128  */


/* declare a structure for complex number representation */

typedef struct {

/* real part */
double real;

/* imag part */
double imag;

} complex;

/* complex is a variable that representas
   complex numbers  (z = real+i*imag)
   for example                              */

/*  define i = sqrt ( -1.0 ) */
/*  real part */

ic.real = 0.0;

/*  imaginary part */

ic.imag = 1.0;

/* Declare a complex number C=3+i*5    */

complex C_number ;
        C_number.real = 3.0;
        C_number.imag = 5.0;
```

```
/* MEMORY Allocation in C programming language */


/* Define constant integer for memory allocation to vectors */

const int nd = 1024;

/* memory to vector x[ ] can be allocated directly as follows:  */

double x[nd];

/* This type of memory allocation is not automatic */

/* Memory is allocated automatically to vector x[ ] with
   malloc command as shown bellow          */

double *x = NULL;
        x = malloc ( nd*sizeof (double ) );

/* Memory allocation for Complex vector variables
   Note that it is double of real variables
   to store both real and imaginary parts           */

complex *fh = NULL;
fh = malloc (2*nd*sizeof (complex));

/* they are used as follows in a for loop */

fh[k].real = C_number.real/nm;
fh[k].imag = C_number.imag/nm;


/* Iteration loops in C for command */

for (i=0; i<nm; i++)
x[i] = 2.0*M_PI*i/nm;

* the number pi in C is M_PI*/

for (k=0; k<nm; k++)
{                                   /*begin loop here*/
kneg = k-nm/2;
C_number.real= C_number.real
             + fh[k].real*cos(kneg*xx)
             - fh[k].imag*sin(kneg*xx);
C_number.imag = C_number.imag
              + fh[k].real*sin(kneg*xx)
              + fh[k].imag*cos(kneg*xx);
}                                   /* end loop 1 here*/
```

```
/* USE OF FOR ITERATION LOOPS WITH IF COMMAND TO
   COMPARE OR TO EXCLUDE CASES */

/* Lagrange interpolation, construct the denominator an if command is
needed to exclude repeated values that meke the denominator zero*/

for (i=0; i<nm; i++) {                  /*begin loop 1*/

pd0=1.0;
for (j=0; j<nm; j++) {                  /*begin loop 2*/

if (i==j) pr = 1.0;

if (i!=j) pr = x[i]-x[j];

pd0 = pd0*pr;

}                                       /* end loop 2*/

pden[i] = pd0;

}                                       /* end loop 1*/


/* Input Output commands in C to write and
   read data from files or from the console*/

/* the fprintf command writes “w” data on file,
                     Appends “a” data on file
                or   reads   “r” data from file */
FILE *fp = NULL;
fp = fopen ("Data.plt","w");
……..……
fprintf (fp,"%d\t %e\t %e\n",k+1, xx, pden[k]);
fclose (fp);

/* the printf command writes on screen  */

printf ("%d\t  %6.5f\t  %6.5f\n", k+1, xx);
```

```
/* How to declare and use FUNCTIONS in C */

/* generic prototype: [data type return] function_name (function
arguments) */
/* All function declarations are put before the beginning of the main
function*/
/* declare a function that computes the area of a square*/
double square_area (double a, double b);
/* Define function square_area.
   Function definitions go after the end of the main function
   or in a separate file.*/
double square_area (double a, double b)
{
     double c;
     c = a*b;
     return c;
}
/* How to use a function in C*/
double a1, b1;
double area;
a1 = 5.0;
b1 = 7.0;
area = square_area (a1, b1);

/* declare a function that computes the dot product of 2 vectors
(ab)*/

double dot_product (double *a, double *b, int n);

/* Define function dot_product */

double dot_product (double *a, double *b, int n) {

      int i;

      double c;

      c = 0.0;
      for (i=0; i<n; i++)
            c = c+a[i]*b[i];

      return c;
}
```

```c
/* declare a function that computes the direct product of 2 vectors
(a:b)*/

void direct_product (double *a, double *b, double *c, int n);

/* Define function dot_product */

void direct_product (double *a, double *b, double *c, int n) {

      int i,j;

      for (i=0; i<n; i++)
            for (j=0; j<n; j++)
                  c[i] = c[i]+a[i]*b[j];
}

/* example dot_product*/
int i;

double *a=NULL;
double *b=NULL;

double dot_prod;

a = malloc (15*sizeof (double ));
b = malloc (15*sizeof (double ));

for (i=0; i<15; i++) {
      a[i] = i+2.0;
      b[i] = i+5.0;
}

dot_prod = dot_product (a,b,15);

/* example direct_product*/
int i;

double *a=NULL;
double *b=NULL;
double *c=NULL;

a = malloc (15*sizeof (double ));
b = malloc (15*sizeof (double ));
c = malloc (15*sizeof (double ));

for (i=0; i<15; i++) {
      a[i] = i+2.0;
      b[i] = i+5.0;
}
direct_product (a,b,c,15);
```