```
/* Math/CS 466/666 Midterm Solutions

   Problem 1(ii). Write or modify a computer program to implement Mueller's method
   and use it to approximate the solution to 1 + z + z^2 + z^3 + z^4 = 0 starting with an
   initial guess of p_0 = 1, p_1 = 2 and p_2 = 3. Print the first 6 iterations of the moethod,
   or in otherwords print p_n for n = 3, ..., 8. */

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

typedef complex double Complex;
Complex f(Complex z){
    // Factor 1 + z + z^2 + z^3 + z^4 for efficiency and accuracy.
    return (((z+1)*z+1)*z+1)*z+1;
}
void printnp(int i,Complex p){
    printf("%3d %24.14e %24.14e\n",i,creal(p),cimag(p));
}

/* The following code, based on Mueler's method given in Burden, Faires and Burden, Numerical Analysis, 10th Edition, Chapter 2.6, Algorithm 2.8, page 97, has been modified
   to perform exactly 6 iterations. */

void mueller(Complex p0,Complex p1,Complex p2,
    double TOL, int N0){
    Complex h1=p1-p0,h2=p2-p1;                  // Step 1
    Complex delta1=(f(p1)-f(p0))/h1;
    Complex delta2=(f(p2)-f(p1))/h2;
    Complex d=(delta2-delta1)/(h2+h1);
    for(int i=3;i<=8;i++){                      // Step 2
        Complex b=delta2+h2*d;                  // Step 3
        Complex D=csqrt(b*b-4*f(p2)*d);
        Complex E;                              // Step 4
        if(cabs(b-D)<cabs(b+D)) E=b+D;
        else E=b-D;
        Complex h=-2*f(p2)/E;                   // Step 5
        Complex p=p2+h,z=f(p);
        printnp(i,p);
        if(cabs(h)<TOL) return(p);              // Step 6
        p0=p1; p1=p2; p2=p;                     // Step 7
        h1=p1-p0; h2=p2-p1;
        delta1=(f(p1)-f(p0))/h1;
        delta2=(f(p2)-f(p1))/h2;
```

```
47          d=(delta2-delta1)/(h2+h1);
48      }
49  }
50
51  int main(){
52      printf(
53          "Math/CS 466/666 Midterm\nProblem 1(ii).\n\n"
54          "%3s %24s %24s\n","n","real(pn)","imag(pn)");
55      Complex p0=1.0,p1=2.0,p2=3.0;
56      printnp(0,p0);
57      printnp(1,p1);
58      printnp(2,p2);
59      mueller(p0,p1,p2,1e-5,15);
60      return 0;
61  }
```