

```

> restart;
>
> horner:= proc()
  local OK, N, II, AA, X0, Y, Z, MM, J, P;
  N:=args[1];
> if type(N,Not(nonnegint)) then
> ERROR("The first argument, giving the degree, must be a nonnegative
  integer");
> fi;
> if nargs<>N+3 then
  ERROR("The number of arguments is not correct");
  fi;
  X0:=args[2];
  if type(X0,Not(complex)) then
  ERROR("The second argument must be a real or complex number");
  fi;
  for II from 3 to N+3 do
  AA[N-II+3] := args[II];
  if type(AA[N-II+3],Not(complex)) then
  ERROR("All arguments except the first must be real or complex numbers");
  fi;
> od;

```

STEP 1

compute b(n) for p(x)

```
> Y := AA[N];
```

compute b(n-1) for q(x) = p'(x)

```
> if N = 0 then
```

```
> Z := 0;
  P:=0;
```

```
> else
```

```
> Z := AA[N];
  P:=Z*x^(N-1);
```

```
> fi;
```

```
> MM := N-1;
```

STEP 2

```
> for II from 1 to MM do
```

```
> J := N-II;
```

compute b(j) for p(x)

```
> Y := Y*X0+AA[J];
  P:=P+Y*x^(J-1);
```

compute b(j-1) for q(x)

```
> Z := Z*X0+Y;
```

```
> od;
```

STEP 3

compute b(0) for p(x)

```
> if N <> 0 then
> Y := Y*X0+AA[0];
> fi;
```

STEP 4

```
> [P,Y,Z];
> end;
```

*horner* := proc()

**local** OK, N, II, AA, X0, Y, Z, MM, J, P;

N := args[1];

**if** type(N, Not(nonnegint)) **then**

    ERROR("The first argument, giving the degree, must be a nonnegative integer" )

**end if**

**if** nargs <> N + 3 **then** ERROR("The number of arguments is not correct") **end if**

X0 := args[2];

**if** type(X0, Not(complex)) **then**

    ERROR("The second argument must be a real or complex number")

**end if**

**for** II **from** 3 **to** N + 3 **do**

    AA[N - II + 3] := args[II];

**if** type(AA[N - II + 3], Not(complex)) **then**

        ERROR("All arguments except the first must be real or complex numbers")

**end if**

**end do**

Y := AA[N];

**if** N = 0 **then** Z := 0; P := 0 **else** Z := AA[N]; P := Z\*x^(N - 1) **end if**

MM := N - 1;

**for** II **to** MM **do** J := N - II; Y := Y\*X0 + AA[J]; P := P + Y\*x^(J - 1); Z := Z\*X0 + Y **end do**

**if** N <> 0 **then** Y := Y\*X0 + AA[0] **end if**

[P, Y, Z]

**end proc**

```
>
```

```
> horner_dir:=proc()
```

```
  printf(`horner returns a list where the first entry\n`);
```

```
  printf(`is the quotient polynomial, the second is the\n`);
```

```
  printf(`value of the original polynomial at the point\n`);
```

```
  printf(`of evaluation, and the third is the value of the derivative\n`);
```

```
  printf(`of the original polynomial at the point of evaluation.\n\n`);
```

```
  printf(`The arguments for horner are, in order:\n`);
```

```
  printf(` (1)the degree of the polynomial\n`);
```

```
  printf(` (2)the point of evaluation (real or complex)\n`);
```

```
  printf(` (3)the coefficients (real or complex) in descending order by
degree\n`);
```

```
  end;
```

```
horner_dir := proc()
```

```
  printf(`horner returns a list where the first entry`);
```

```
  printf(`is the quotient polynomial, the second is the`);
```

```
  printf(`value of the original polynomial at the point`);
```

```
  printf(`of evaluation, and the third is the value of the derivative`);
```

```
  printf(`of the original polynomial at the point of evaluation.`);
```

```
  printf(`The arguments for horner are, in order:`);
```

```
printf` (1)the degree of the polynomial );  
printf` (2)the point of evaluation (real or complex) );  
printf` (3)the coefficients (real or complex) in descending order by degree )  
end proc
```

```
>
```