

문제1

고차방정식의 해를 구하기 위해 주어진 x 의 범위를 단위별로 대입해 결과를 얻어내고 그렇게 나온 결과 중 내가 원하는 결과에 가장 가까운 x 를 찾아낸 후 그 사이에 더 범위를 세분화하면서 대입을 반복하다보면 내가 원하는 결과에 근사하는 값을 가지는 x 를 알아낼 수 있을 것이라고 생각했다. 그러기 위하여 선형 리스트 구조체를 만들어 $f(x)$ 를 구성하고, x 값을 대입해 결과를 관찰하는 함수를 따로 설계하였다.

먼저 $f(x)=0$ 을 만족하는 x 와 $f(x)$ 가 최속= a 되는 x 값을 만족하기 위해 아래 결과와 같이 선형리스트로 함수 $f(x)$ 를 만들고 x 범위 $[-10,10]$ 에서 0.5씩 증가시켜가며 값을 결과를 관찰하였다. 그 결과 -10 부터 -0.5까지 꾸준히 값이 감소 했으며 그 이후에 다시 증가하는 양상의 그래프임을 알 수 있었다. $x=-1.5$ 일때 약 17 이고 $x=-1$ 일때 -0.5 였고, $x=0.5$ 일때 약 -0.93 이고 $x=1$ 일때 7.9 였으므로 이 사이의 범위를 더 세분화하여 대입하면 $f(x)$ 가 0이 되는 지점을 찾을 수 있을 것이라고 생각했다. 또한 $x=-1.0\sim 0.0$ 에서 각각 약 -0.5, -2.73125, -2.0 이었다. 즉 감소하다가 증가하는 형태인데 이 근방의 x 값들을 더 세분화하여 더 작은 값이 있는지 추가로 확인하기로 했다.

```
지수:4 계수:6.700000
지수:3 계수:3.200000
지수:2 계수:-1.000000
지수:1 계수:1.000000
지수:0 계수:-2.000000

f(x)에 -10.000000를 대입한 값은 63688.000000입니다.
f(x)에 -9.500000를 대입한 값은 51726.568750입니다.
f(x)에 -9.000000를 대입한 값은 41533.900000입니다.
f(x)에 -8.500000를 대입한 값은 32926.468750입니다.
f(x)에 -8.000000를 대입한 값은 25730.800000입니다.
f(x)에 -7.500000를 대입한 값은 19783.468750입니다.
f(x)에 -7.000000를 대입한 값은 14931.100000입니다.
f(x)에 -6.500000를 대입한 값은 11030.368750입니다.
f(x)에 -6.000000를 대입한 값은 7948.000000입니다.
f(x)에 -5.500000를 대입한 값은 5560.768750입니다.
f(x)에 -5.000000를 대입한 값은 3755.500000입니다.
f(x)에 -4.500000를 대입한 값은 2429.068750입니다.
f(x)에 -4.000000를 대입한 값은 1488.400000입니다.
f(x)에 -3.500000를 대입한 값은 850.468750입니다.
f(x)에 -3.000000를 대입한 값은 442.300000입니다.
f(x)에 -2.500000를 대입한 값은 200.968750입니다.
f(x)에 -2.000000를 대입한 값은 73.600000입니다.
f(x)에 -1.500000를 대입한 값은 17.368750입니다.
f(x)에 -1.000000를 대입한 값은 -0.500000입니다.
f(x)에 -0.500000를 대입한 값은 -2.731250입니다.
f(x)에 0.000000를 대입한 값은 -2.000000입니다.
f(x)에 0.500000를 대입한 값은 -0.931250입니다.
f(x)에 1.000000를 대입한 값은 7.900000입니다.
f(x)에 1.500000를 대입한 값은 41.968750입니다.
f(x)에 2.000000를 대입한 값은 128.800000입니다.
f(x)에 2.500000를 대입한 값은 305.968750입니다.
f(x)에 3.000000를 대입한 값은 621.100000입니다.
f(x)에 3.500000를 대입한 값은 1131.868750입니다.
f(x)에 4.000000를 대입한 값은 1906.000000입니다.
f(x)에 4.500000를 대입한 값은 3021.268750입니다.
f(x)에 5.000000를 대입한 값은 4565.500000입니다.
f(x)에 5.500000를 대입한 값은 6636.568750입니다.
f(x)에 6.000000를 대입한 값은 9342.400000입니다.
f(x)에 6.500000를 대입한 값은 12800.968750입니다.
f(x)에 7.000000를 대입한 값은 17140.300000입니다.
f(x)에 7.500000를 대입한 값은 22498.468750입니다.
f(x)에 8.000000를 대입한 값은 29023.600000입니다.
f(x)에 8.500000를 대입한 값은 36873.868750입니다.
f(x)에 9.000000를 대입한 값은 46217.500000입니다.
f(x)에 9.500000를 대입한 값은 57232.768750입니다.
f(x)에 10.000000를 대입한 값은 70108.000000입니다.
D:\#단국대학교2021-2#자료구조과제3#Debug과제3.exe(프로세스 3880개)이(가
```



여러 번 반복한 결과 출력 화면을 첨부하진 못했지만 다음과 같은 결과를 얻어낼 수 있었다.

$f(x)$ 에 -1.040000를 대입한 값은 0.116888입니다.

$f(x)$ 에 -1.030000를 대입한 값은 -0.046717입니다.

$f(x)$ 에 -1.033000를 대입한 값은 0.001690입니다. (0에 가장 근접)

$f(x)$ 에 -1.032000를 대입한 값은 -0.014509입니다.

$f(x)$ 에 0.620000를 대입한 값은 -0.011736입니다.

$f(x)$ 에 0.630000를 대입한 값은 0.088699입니다.

$f(x)$ 에 0.621000를 대입한 값은 -0.001878입니다.

$f(x)$ 에 0.622000를 대입한 값은 0.008021입니다.

또한 $x = -0.59$ 에서 -2.783450으로 **최소값**에 가까운 값을 얻어낼 수 있었다.

f(x)에	-0.720000	대입하여 기울기가 0인 x의 근사값을 찾고 그 x를 미분전 함수에 대입하며 실제 최솟값인지 확인하는 방법(기울기가 0이라고 해서 항상 그래프가 아래쪽에 있는 것은 아니기 때문)으로 더 개선된 결과를 얻어낼 수도 있을 것 같다고 생각했다.	-2.692243입니다.
f(x)에	-0.710000		-2.656833입니다.
f(x)에	-0.700000		-2.678930입니다.
f(x)에	-0.690000		-2.698632입니다.
f(x)에	-0.680000		-2.716030입니다.
f(x)에	-0.670000		-2.731216입니다.
f(x)에	-0.660000		-2.744280입니다.
f(x)에	-0.650000		-2.755308입니다.
f(x)에	-0.640000		-2.764387입니다.
f(x)에	-0.630000		-2.771602입니다.
f(x)에	-0.620000		-2.777035입니다.
f(x)에	-0.610000		-2.780768입니다.
f(x)에	-0.600000		-2.782880입니다.
f(x)에	-0.590000		-2.783450입니다.
f(x)에	-0.580000		-2.782553입니다.
f(x)에	-0.570000		-2.780266입니다.
f(x)에	-0.560000		-2.776660입니다.
f(x)에	-0.550000		-2.771808입니다.
f(x)에	-0.540000		-2.765780입니다.
f(x)에	-0.530000		-2.758644입니다.
f(x)에	-0.520000		-2.750467입니다.
f(x)에	-0.510000		-2.741315입니다.
f(x)에	-0.500000		-2.731250입니다.
f(x)에	-0.490000		-2.720335입니다.
f(x)에	-0.480000		-2.708631입니다.
f(x)에	-0.470000		-2.696195입니다.
f(x)에	-0.460000		-2.683086입니다.
f(x)에	-0.450000		-2.668358입니다.
f(x)에	-0.440000		-2.655066입니다.
f(x)에	-0.430000		-2.640263입니다.

시간이 없어 x를 대입하는 것으로 2,3번을 해결했지만 f(x)를 비문하여 기울기가 0인 x의 근사값을 찾고 그 x를 미분전 함수에 대입하며 실제 최솟값인지 확인하는 방법(기울기가 0이라고 해서 항상 그래프가 아래쪽에 있는 것은 아니기 때문)으로 더 개선된 결과를 얻어낼 수도 있을 것 같다고 생각했다.

문제2

동적으로 원형큐를 생성하고 문자를 넣을 수 있도록 설계했다. 큐 생성 및 할당, 삽입, 삭제, 출력, 공간 여부 확인 등의 함수를 모두 독립적으로 작성했다. isEmpty()를 통해 원형큐에 공간이 있는지 알 수 있지만 삽입, 삭제 중 큐에 공간이 없거나 혹은 삭제할 원소가 없을 때도 메시지를 출력한다.

```

선택 Microsoft Visual Studio 디버그 콘솔
nat
원형큐에 공간이 있습니다.
enqueue:A 를 원형큐에 넣습니다.
enqueue:B 를 원형큐에 넣습니다.
enqueue:C 를 원형큐에 넣습니다.
현재 원형큐의 모습입니다.
ABC
dequeue:A 를 원형큐에서 뺍니다.
dequeue:B 를 원형큐에서 뺍니다.
현재 원형큐의 모습입니다.
C
dequeue:C 를 원형큐에서 뺍니다.
현재 원형큐의 모습입니다.

dequeue불가: 원형큐가 비어있습니다.

D:\#단국대학교\2021-2\자료구조\과제3\Debug\과제3.exe(프로세스 15900개)이(가) 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요...

```

문제3

'(' 를 만나면 스택에 해당 인덱스를 넣고 ')' 를 만나면 가장 '(' 와 ')' 인덱스를 출력한다. 가장 늦게 들어간 '('가 가장 먼저 만난 ')'와 쌍을 이루기 때문에 스택의 top을 출력해주면 된다. 매 순간마다 스택의 상태(결과)를 출력해줘서 중첩 '(' ')' 가 어떻게 스택에 쌓여 있는지 확인할 수 있었다.

Microsoft Visual Studio 디버그 콘솔

```
0
0 1
0 1 2
0 1 2
0 1 2
0 1 2
0 1 2
(2,6)
0 1
0 1
0 1
0 1
0 1
0 1
0 1
(1,13)
0
0
0 15
0 15
0 15
0 15
(15,19)
0
0
0 21
0 21
0 21
0 21
(21,25)
0
0
0 27
0 27
0 27
0 27
(27,31)
0
(0,32)
34
34
34
34
(34,38)
0
0
0
0
```

```
0
0
(0,6)
10
10
10
10 13
10 13
10 13
10 13
(13,17)
10
10
10
(10,20)
22
22
22
22
(22,26)
D:\#단국대학교#2021-2#자료구조
이 창을 닫으려면 아무 키나 누르
```

줄단위로 스택을 출력(가장 오른쪽이 top)하고 각 pair 순서쌍을 출력했다. (공백은 스택에 아무것도 없는 경우이다.)

문제4

연산자의 우선순위를 고려하여 infix 식을 postfix 식으로 변환하기 위해 스택을 사용하였다.

해당 식 5개를 가지고 테스트 해본 결과이다.

```
char ex1[] = "(A+B)*C-D";  
char ex2[] = "A*B*C/D+E+F*G";  
char ex3[] = "A/B+(C*D)-E";  
char ex4[] = "A*(B+C)-E/F";  
char ex5[] = "A+B+C+D+E";
```

```
AB+C*D-  
AB*C*D/E+FG*+  
AB/CD*+E-  
ABC+*EF/-  
AB+C+D+E+
```

```
D:##단국대학교##2021-2##자료구조##과제3##Debug##과제3.exe(프  
이 창을 닫으려면 아무 키나 누르세요...
```