REPORT

과 목 명 : 자료구조
담담교수 : 황두성 교수님
소　　속 : 소프트웨어학과
학　　번 : 32151671
이　　름 : 박민혁

단국대학교
Dankook University

문제 1.

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "functions.h"

int main(void)
{
double n;
Listheader mlist;
init(&mlist);
insertnode(&mlist, 6.7, 7);
insertnode(&mlist, 3.2, 3);
insertnode(&mlist, -1, 2);
insertnode(&mlist, 1, 1);
insertnode(&mlist, -4, 0);


for (n = 0.1; n < 1.1;)
{
printf("n의값 %.1f", n);
printf("계산값 %.3f\n\n", cacl(&mlist, n));
n += 0.1;
}
return 0;
}
```

functions.h

```
doble ca치(Listheader *plist, double n);
void init(Listheader *plist);
void insertnode(Listheader *plist, double cf, int ind);
void polyprint(Listheader *plist);
```

functions.c

```
typedef struct ListNode
{
double cf;
int ind;
struct ListNode* link;

}ListNode
```

```
typedef struct Listheader
{
int length;
ListNode* head;
ListNode* tail;
}Listheader

void init(Listheader* plist)
{
plist->length = 0;
plist->head = plist->tail = NULL
}

void insertnode(Listheader* plist, double cf, int ind)
{
ListNode* node = (ListNode*)malloc(sizeof(ListNode));
node->cf = cf
node->ind = ind
node->link = NULL

if (plist == NULL) return

if (plist->head == NULL)
{
plist->head = node;
plist->tail = node;
}

else
{
plist->tail->link = node;
plist->tail = node;
}
plist->length++;
}
void polyprint(Listheader* plist)
{
ListNode* p = plist->head;
printf("F(x) = ");
for (; p != NULL p = p->link) {
printf(" %0.2fX^%d + ", p->cf, p->ind);
}
printf("\n\n");

}
```

```
double cacl(Listheader* plist, double n)
{
ListNode* p = plist->head;
double sum = 0;
for (; p != NULL p = p->link)
{
sum += (p->cf)*(pow(n, p->ind));
}
return sum;
}
```

MakeFile

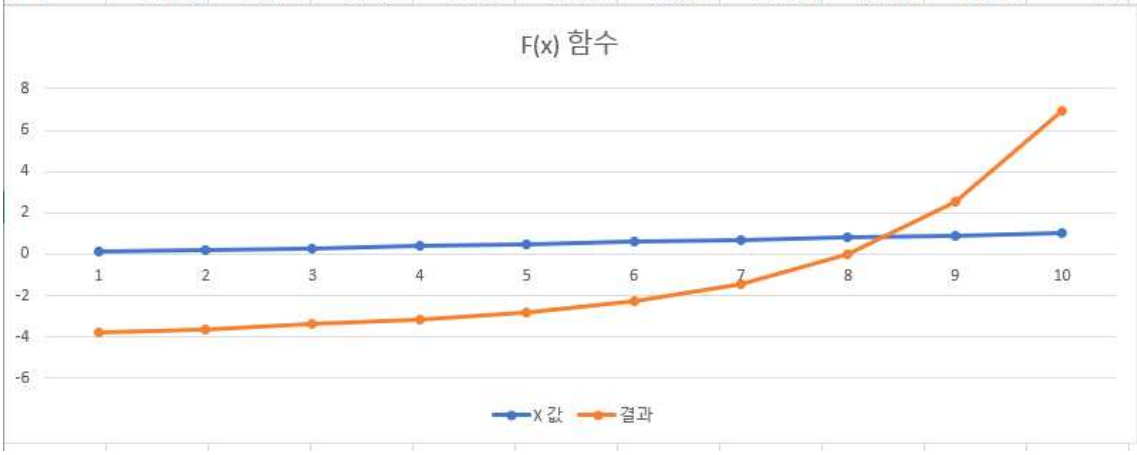
```
all : main
main : main.o functions.o
        gcc -o main main -o functions,o
main.o : main.c functions.h
        gcc -c main.c
functions.o : functions.c functions..h
        gcc -c functions.c
```

문제 1. 결과창

```
선택 C:₩WINDOWS₩system32₩cmd.exe
F(x) =  6.70X^7 +  3.20X^3 +  -1.00X^2 +  1.00X^1 +  -4.00X^0 +
x의 값 : 0.1   계산 값 : -3.9068
x의 값 : 0.2   계산 값 : -3.8143
x의 값 : 0.3   계산 값 : -3.7021
x의 값 : 0.4   계산 값 : -3.5442
x의 값 : 0.5   계산 값 : -3.2977
x의 값 : 0.6   계산 값 : -2.8812
x의 값 : 0.7   계산 값 : -2.1406
x의 값 : 0.8   계산 값 : -0.7965
x의 값 : 0.9   계산 값 : 1.6274
x의 값 : 1.0   계산 값 : 5.9000
계속하려면 아무 키나 누르십시오 . . .
```

문제 1. 그래프

X 값	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
결과	-3.9068	-3.8143	-3.7021	-3.5442	-3.2977	-2.8812	-2.1406	-0.7965	1.6274	5.9



문제 2

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "functions.h"

int main(void)
{
queue q;
init(&q);
int t = 0;
int select = 0;
printf(" 1 = 삽입 2 = 삭제 3 = 현재큐출력 = 프로그램종료");
while (1)
{
printf(" 수행할작업을선택");
scanf_s("%d", &select);
if (select == 1)
{
printf("수입력 ");
scanf_s("%d", &t);
enqueue(&q, t);
}
else if (select == 2)
{
printf("Deleted Data : ");
printf("%d \n", dequeue(&q));
}
else if (select == 3)
{
qprint(&q);
}
else if (select == 4)
break
}
printf("\n");
return 0;
}
```

functions.c

```
#include “functions.h”
#define TRUE  1
#define FALSE 0
#define QL  20

typedef int data

typedef struct cqueue
{
int front;
int rear;
data quearr[QL];
}cqueue

typedef cqueue queue

void init(queue* pq)
{
pq->front = 0;
pq->rear = 0;
for (int i = 0; i < QL i++)
{
pq->quearr[i] = -1;
}
}

int empty(queue* pq)
{
if (pq->rear == pq->front)
return TRUE
else
return FALSE
}

int next(int n)
{
if (n == QL - 1)
return 0;
else
return n + 1;
}

void enqueue(queue* pq, data data)
{
```

```
if (next(pq->rear) == pq->front)
{
printf("꽉차있다");
exit(-1);
}
pq->rear = next(pq->rear);
pq->quearr[pq->rear] = data
}

data dequeue(queue* pq)
{
if (empty(pq))
{
printf("비어있네\n");
exit(-1);
}

pq->front = next(pq->front);
return pq->quearr[pq->front];
}

data qpeek(queue* pq)
{
if (empty(pq))
{
printf("비어있네\n");
exit(-1);
}

return pq->quearr[(next(pq->front))];
}

void qprint(queue* pq)
{
printf("사용된적이없는값은1 입니다 \n");
for (int i = 0; i < QL i++)
{
printf("%d ", pq->quearr[i]);
}
printf("\n");
}
```

functions.h

```
void qprint(queue *pq);
Data qpeek(queue * pq);
Data dequeue(queue * pq);
void enqueue(queue * pq, data data);
int next(int n);
int empty(queue * pq);
void init(queue *pq);
```

Makefile

```
all : main
main : main.o functions.o
        gcc -o main main -o functions,o
main.o : main.c functions.h
        gcc -c main.c
functions.o : functions.c functions..h
        gcc -c functions.c
```

문제 2. 결과

```
C:₩WINDOWS₩system32₩cmd.exe

1 = 삽입, 2 = 삭제, 3 = 현재 큐 출력 4 = 프로그램 종료
 수행 할 작업을 선택 :1
수 입력 : 234
 수행 할 작업을 선택 :1
수 입력 : 234
 수행 할 작업을 선택 :1
수 입력 : 22
 수행 할 작업을 선택 :1
수 입력 : 33
 수행 할 작업을 선택 :2
Deleted Data : 234
 수행 할 작업을 선택 :2
Deleted Data : 234
 수행 할 작업을 선택 :2
Deleted Data : 22
 수행 할 작업을 선택 :3
사용된 적이 없는 값은 -1 입니다.
 -1  234  234  22  33  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1

 수행 할 작업을 선택 :2
Deleted Data : 33
 수행 할 작업을 선택 :3
사용된 적이 없는 값은 -1 입니다.
 -1  234  234  22  33  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1

 수행 할 작업을 선택 :
```

```
 수행 할 작업을 선택 :1
수 입력 : 234
 수행 할 작업을 선택 :1
수 입력 : 55
 수행 할 작업을 선택 :1
수 입력 : 33
 수행 할 작업을 선택 :3
사용된 적이 없는 값은 -1 입니다.
 -1  23  33  55  236  22  345  32  235  234  23455  234  55  33  -1  -1  -1  -1  -1  -1

 수행 할 작업을 선택 :1
수 입력 : 22
 수행 할 작업을 선택 :1
수 입력 : 55
 수행 할 작업을 선택 :1
수 입력 : 666
 수행 할 작업을 선택 :1
수 입력 : 2
 수행 할 작업을 선택 :1
수 입력 : 3
 수행 할 작업을 선택 :1
수 입력 : 2
 수행 할 작업을 선택 :3
사용된 적이 없는 값은 -1 입니다.
 -1  23  33  55  236  22  345  32  235  234  23455  234  55  33  22  55  666  2  3  2

 수행 할 작업을 선택 :1
수 입력 : 5
Queue is Full !
계속하려면 아무 키나 누르십시오 . . .
```

문제 3.

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
matrix_header M;
init(&M);

int **Matrix=NULL
int i, j;
int count=0;
int Mrow, Mcol;

srand((unsigned)time(NULL));
printf("크기");
scanf_s("%d %d", &Mrow, &Mcol);

Matrix = (int **)malloc(sizeof(int*)*Mrow);
for (i = 0; i < Mrow; i++)
Matrix[i] = (int*)malloc(sizeof(int)*Mcol);

for (i = 0; i < Mrow; i++)
{
for (j = 0; j < Mcol; j++)
{
if (count < (3 * i*j) / 10)
{
Matrix[i][j] = rand() % 101;
count++;

}
else
Matrix[i][j] = 0;

}
}
printf("\nMatrix\n\n");
for (i = 0; i < Mrow; i++)
{
printf("——);
```

```
for (j = 0; j < Mcol; j++)
{
printf("%2d", Matrix[i][j]);
}
printf("——");
printf("\n");
}

for (i = 0; i < Mrow; i++)
{
for (j = 0; j < Mcol; j++)
{
if (Matrix[i][j] != 0)
{
insertmatrix(&M, i, j, Matrix[i][j]);
}
}
}
printf("\n희소행렬로바꾼결과");
printf("||rowcolvalue||\n");
printf("||%d%d%d", Mrow, Mcol, count);
print_matrix(&M);
printf("sum =%d", cacl(&M));
for (i = 0; i < Mrow; i++)
{
free(Matrix[i]);
}
free(Matrix);
return 0;
}
```

functions.c

```
#include "functions.h"
typedef struct matrix_node
{
int row;
int col;
int value;
struct matrix_node* link;
}matrix_node

typedef struct matrix_header
{
matrix_node* head;
```

```
matrix_node* tail;
}matrix_header

void init(matrix_header* m)
{
m->head = m->tail = NULL
}

void insertmatrix(matrix_header* m, int row, int col, int value)
{
matrix_node* node = (matrix_node*)malloc(sizeof(matrix_node));
node->row = row
node->col = col
node->value = value
node->link = NULL

if (m == NULL)
return

if (m->head == m->tail)
{
m->head = node;
m->tail = node;
}
else
{
m->tail->link = node;
m->tail = node;
}
}

void print_matrix(matrix_header* m)
{
matrix_node *m1=m->head;
for (; m1 != NULL m1 = m1->link)
{
printf("|| %d      %d      %d||\n", m1->row, m1->col, m1->value);
}
printf("\n\n");
}

int cacl(matrix_header* m)
{
int sum = 0;
matrix_node* m1 = m->head;
```

```
for (; m1 != NULL m1 = m1->link)
{
sum = sum + (m1->value);
}
return sum;
}

        return sum;
}
```

functions.h

```
void init(matrix_header *m);
void insertmatrix(matrix_header *m, int row, int col, int value);
void print_matrix(matrix_header *m) ;
int sum(matrix *m);
```

Makefile

```
all : main
main : main.o functions.o
        gcc -o main main -o functions,o
main.o : main.c functions.h
        gcc -c main.c
functions.o : functions.c functions..h
        gcc -c functions.c
```

문제 3. 결과

선택 C:₩WINDOWS₩system32₩cmd.exe

```
M1 크기 : 10 10

M1

--  0   0   0   0   0   0   0   0   0   0 --
--  0   0   0   0  56   0   0  59   0   0 --
--  0   0   0   0   0  15   0  62   0  90 --
--  0   0   0   0   0   0   0   5   0  80 --
--  0   0   0   0   0   0   0   0  87  72 --
--  0   0   0   0   0   0   0   0  14  59 --
--  0   0   0   0   0   0   0   0  44  80 --
--  0   0   0   0   0   0   0   0  25  68 --
--  0   0   0   0   0   0   0   0  99  17 --
--  0   0   0   0   0   0   0   0  24  15 --

 희소 행렬로 바꾼 결과
|| row  col  value   ||
||  10   10    20    ||
||  1   4    56    ||
||  1   7    59    ||
||  2   5    15    ||
||  2   7    62    ||
||  2   9    90    ||
||  3   7    5    ||
||  3   9    80    ||
||  4   8    87    ||
||  4   9    72    ||
||  5   8    14    ||
||  5   9    59    ||
||  6   8    44    ||
||  6   9    80    ||
||  7   8    25    ||
||  7   9    68    ||
||  8   8    99    ||
||  8   9    17    ||
||  9   8    24    ||
||  9   9    15    ||


 sum of value = 971

계속하려면 아무 키나 누르십시오 . . .
```

문제 4.

main.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include “functions.h”

int main(int argc, char *argv[])
{
        char exp1[] = "a+b+c+d+e+f";
        char exp2[] = "a*b+c/d+e-f+g";
        char exp3[] = "a/b+(c*d)-e";
        char exp4[] = "a-(b+c)*d*e";
        char exp5[] = "a*b*c/d+e+f*g";

        printf("입력한 Infix \n\n");
        puts(exp1);
        puts(exp2);
        puts(exp3);
        puts(exp4);
        puts(exp5);

        printf("\nProfix 출력 \n\n");
        Change(exp1);
        printf("\n");
        Change(exp2);
        printf("\n");
        Change(exp3);
        printf("\n");
        Change(exp4);
        printf("\n");
        Change(exp5);
        printf("\n");
        return 0;
}
```

functions.c

```
#include “functions.h”
struct Stack
```

```
{
        int top;
        unsigned Capa;
        int* array;
};

struct Stack* MakeStack(unsigned Capa)
{
        struct Stack* stack = (struct Stack*) malloc(sizeof(struct Stack));

        if (!stack)
                return NULL;

        stack->top = -1;
        stack->Capa = Capa;

        stack->array = (int*)malloc(stack->Capa * sizeof(int));

        if (!stack->array)
                return NULL;
        return stack;
}
int isEmpty(struct Stack* stack)
{
        return stack->top == -1;
}
char peek(struct Stack* stack)
{
        return stack->array[stack->top];
}
char pop(struct Stack* stack)
{
        if (!isEmpty(stack))
                return stack->array[stack->top--];
        return '$';
}
void push(struct Stack* stack, char op)
{
        stack->array[++stack->top] = op;
}
```

```
int isOperand(char ch)
{
	return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z');
}
int Prio(char ch)
{
	switch (ch)
	{
	case '+':
	case '-':
		return 1;

	case '*':
	case '/':
		return 2;

	case '^':
		return 3;
	}
	return -1;
}
int Change(char* exp)
{
	int i, k;

	struct Stack* stack = MakeStack(strlen(exp));
	if (!stack)
		return -1;

	for (i = 0, k = -1; exp[i]; ++i)
	{
		if (isOperand(exp[i]))
			exp[++k] = exp[i];

		else if (exp[i] == '(')
			push(stack, exp[i]);

		else if (exp[i] == ')')
		{
			while (!isEmpty(stack) && peek(stack) != '('
```

```
                                        exp[++k] = pop(stack);
                                if (!isEmpty(stack) && peek(stack) != '(')
                                        return -1;
                                else
                                        pop(stack);
                        }
                        else
                        {
                                while (!isEmpty(stack) && Prio(exp[i]) <= Prio(peek(stack)))
                                        exp[++k] = pop(stack);
                                push(stack, exp[i]);
                        }

                }

                while (!isEmpty(stack))
                        exp[++k] = pop(stack);

                exp[++k] = '\0';
                printf("%s", exp);
}
```

functions.h

```
struct Stack* MakeStack(unsigned Capa);
int Change(char* exp);
int Prio(char ch);
int isOperand(char ch);
void push(struct Stack* stack, char op);
char pop(struct Stack* stack);
char peek(struct Stack* stack);
int isEmpty(struct Stack* stack);
struct Stack* MakeStack(unsigned Capa);
```

Makefile

```
all : main
main : main.o functions.o
        gcc -o main main -o functions,o
main.o : main.c functions.h
        gcc -c main.c
functions.o : functions.c functions..h
        gcc -c functions.c
```

문제 4. 결과

```
C:₩WINDOWS₩system32₩cmd.exe
입력한 Infix

a+b+c+d+e+f
a*b+c/d+e-f+g
a/b+(c*d)-e
a-(b+c)*d*e
a*b*c/d+e+f*g

Profix 출력

ab+c+d+e+f+
ab*cd/+e+f-g+
ab/cd*+e-
abc+d*e*-
ab*c*d/e+fg*+
계속하려면 아무 키나 누르십시오 . . .
```