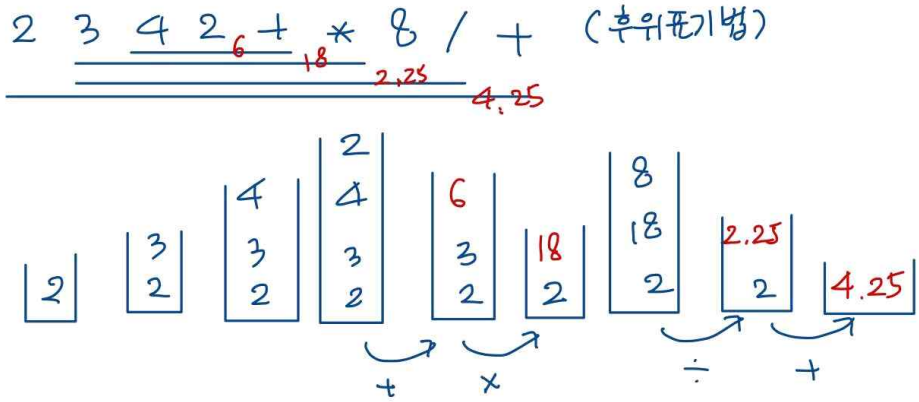
자료구조 5주차 과제

제출일: 2022-10-16
학번/이름: 2016121150 / 윤준영

문1) 교재의 calc 함수를 사용하여 후위 표기법 식 "2 3 4 2 + * 8 / +"를 계산하려고 한다. 스택의 변화 과정을 모두 쓰시오. (2점)



문2) 다음 프로그램을 사용하여 4 + 5 * ((5 - 3)*2)를 후위 표기법으로 변환하려 한다. 4장 교재 24쪽의 실행 예와 같이, 실행 시 스택의 변화와 출력을 제시하시오. (3점)

```
void postfix(char *dst, char *src){
    char c;
    init_stack();
    while (*src) {
       if (*src== '(') {
           push(*src); src++;
       }
       else if (*src== ')') {
           while (get_stack_top() != '(') {
       *dst++ = pop(); *dst++ = ' ';
           }
           pop(); src++;
       }
       else if (is_operator(*src)) {
           while (!is_stack_empty() &&
             precedence(get_stack_top())>= precedence(*src)){
             *dst++ = pop(); *dst++ = ' ';
           }
           push(*src); src++;
       }
       else if (*src>= '0' && *src<= '9') {
           do {
              *dst++ = *src++;
           } while (*src>= '0' && *src<= '9');
```

```
            *dst++ = ' ';
        }
        else src++;
    }
    while (!is_stack_empty()) {
        *dst++ = pop(); *dst++ = ' ';
    }
    dst--; *dst= 0;
}
```

4 + 5 * ((5 - 3) * 2)

입력	출력	스택 (상단 → 하단)
4	4	
+	4	+
5	4 5	+
*	4 5	* +
(	4 5	(* +
(	4 5	((* +
5	4 5 5	((* +
-	4 5 5	- ((* +
3	4 5 5 3	- ((* +
)	4 5 5 3 -	(* +
*	4 5 5 3 -	* (* +
2	4 5 5 3 - 2	* (* +
)	4 5 5 3 - 2 *	* +
끝	4 5 5 3 - 2 * * +	

문3) 다음은 배열로 구현한 큐를 사용하는 프로그램이다. 이 프로그램을 배열 대신 연결리스트를 사용하는 방식으로 변경하시오. 단 main 함수는 변경하지 않고 그대로 사용해야 한다. (5점)

HW5 > C hw5_3_.c > ...

```c
/* 연결리스트 사용 프로그램 */
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

typedef struct _node *nodeptr;
typedef struct _node {
        int val;
        nodeptr next;
} node;

typedef struct {
        nodeptr head;
        nodeptr tail;
        nodeptr rear;
} queue;

queue * create_queue() {
    queue *q = (queue *)malloc(sizeof(queue));
    q->head = (node *)malloc(sizeof(node));
    q->tail = (node *)malloc(sizeof(node));
    q->rear = q->head;
    // q->head->next 가 q->tail이면 empty
    q->head->next = q->tail;
    return q;
}

void clear_queue(queue *q) {
    node *p;
    while (1){
        p = malloc(sizeof(node));
        p = q->head->next;
        if (p==q->tail) break;
        q->head->next = p->next;
        free(p);
    }
    q->rear = q->head;
    q->head->next = q->tail;
}

int put(queue *q, int k)
{
    nodeptr p;
    p = malloc(sizeof(node));
    q->rear->val = k;
    p->next = q->tail;
    q->rear->next = p;
    q->rear = p;
    return k;
}

int get(queue *q){
    int i;
    node *p;
    if (q->head->next == q->tail){
        printf("\n      Queue underflow.");
        return -1;
    }
    i = q->head->val;
    p = q->head;
    q->head = q->head->next;
    free(p);
    return i;
}

void print_queue(queue *q)  {
    node *p;
    printf("\n  Queue contents : Front ----> Rear\n");
    for (p = q->head; p->next != q->tail; p = p->next)
        printf("%-6d", p->val);
}
```

작성되어 있는 함수가 이중연결리스트가 아니기 때문에 value와 다음 노드포인터 next만을 가지는 노드를 통해 연결리스트를 구현하였다. 또한 create_queue함수가 rear을 head로 설정하였기 때문에 head 노드에도 value를 가지게 구현하였다.

clear_queue : tail노드가 될 때까지 malloc과 free를 반복하여 메모리 할당을 해제한다. 마지막으로 rear노드를 head노드로 설정한다. head의 next 노드는 tail로 설정한다. 이렇게 설정하면 head 노드의 값은 남아있지만 print_queue나 get할 때에 빈 큐로 인식하여 사용되지 않는 값이 되고, put함수를 사용할 때 덮어씌워지게 된다.

put: 먼저 새로이 rear가 될 노드 p를 생성하였다. 현재 q의 rear의 값을 k로 설정한 후, 새로운 rear가 될 p의 next를 tail로 설정하였다. 그 후 rear의 next를 새로운 rear가 될 p로 설정하고, p가 q의 rear가 되게 설정하였다. head->...->rear(old, val=k)->rear(new, p)->tail이 된다.

get: 먼저 빈 큐인지 확인한 후, 빈 큐이면 -1을 return한다. 그 후 front 노드를 head 노드로

설정하였기 때문에, 반환할 i를 head->val로 설정한다. 이제 head는 메모리 할당을 해제하여야 하므로 free해주기 위해 p로 설정한다. 새롭게 head 노드가 될 노드는 그 다음 노드이기 때문에 q->head=q->head->next를 통해 새로운 head 노드를 지정한 후, 이전 head 노드를 free한다.

print_queue: for문을 사용하여 head 노드부터 다음 노드를 순차적으로 tail 전 노드가 될 때까지 print한다.

```
73  void main(void){
74      int i;
75      queue *q = create_queue();
76      printf("\nPut5, 4, 7, 8, 2, 1");
77      put(q, 5); put(q, 4); put(q, 7);
78      put(q, 8); put(q, 2); put(q, 1);
79      print_queue(q);
80      printf("\nGet"); i= get(q);
81      print_queue(q);
82      printf("\n   getting value is %d", i);
83      printf("\nPut3, 2, 5, 7");
84      put(q,3); put(q, 2); put(q, 5); put(q, 7);
85      print_queue(q);
86      printf("\nNowqueue is full, put 3");
87      put(q, 3);
88      print_queue(q);
89      printf("\nInitializequeue");
90      clear_queue(q);
91      print_queue(q);
92      printf("\nNowqueue is empty, get");
93      i= get(q);
94      print_queue(q);
95      printf("\n   getting value is %d\n", i);
96  }
```

main 함수를 같은 함수를 사용하였다.

좌측은 배열을 사용하여 큐를 구현하여 실행한 결과이고, 우측은 연결리스트를 사용한 큐를 구현한 결과이다. 같은 결과를 출력하는 것을 알 수 있다.

```
Put5, 4, 7, 8, 2, 1
  Queue contents : Front ----> Rear
5    4    7    8    2    1
Get
  Queue contents : Front ----> Rear
4    7    8    2    1
   getting value is 5
Put3, 2, 5, 7
  Queue contents : Front ----> Rear
4    7    8    2    1    3    2    5    7
Nowqueue is full, put 3
  Queue contents : Front ----> Rear
4    7    8    2    1    3    2    5    7    3
Initializequeue
  Queue contents : Front ----> Rear

Nowqueue is empty, get
    Queue underflow.
  Queue contents : Front ----> Rear

   getting value is -1
```

```
Put5, 4, 7, 8, 2, 1
  Queue contents : Front ----> Rear
5    4    7    8    2    1
Get
  Queue contents : Front ----> Rear
4    7    8    2    1
   getting value is 5
Put3, 2, 5, 7
  Queue contents : Front ----> Rear
4    7    8    2    1    3    2    5    7
Nowqueue is full, put 3
  Queue contents : Front ----> Rear
4    7    8    2    1    3    2    5    7    3
Initializequeue
  Queue contents : Front ----> Rear

Nowqueue is empty, get
      Queue underflow.
  Queue contents : Front ----> Rear

   getting value is -1
```