OS Assignment 1

Name：刘乐奇

sid：12011327

part I

1. A
2. D
3. C
4. C
5. C

part II

Finished on OJ

A. Minimum Difference

```c
#include <stdio.h>
#include <stdlib.h>
#define ll long long

ll res[2000010];
typedef struct link_node *node_ptr;
typedef struct link_node
{
    ll val;
    int rank;
    node_ptr prev;
    node_ptr next;
} node, *node_ptr;

int comp(const void *p1, const void *p2)
{
    return ((**(node_ptr *)p1).val - (**(node_ptr *)p2).val);
}

int main()
{
    int n;
    scanf("%d\n", &n);
    node_ptr *arr = (node_ptr *)malloc(sizeof(node_ptr) * n);
    node_ptr *to_arr = (node_ptr *)malloc(sizeof(node_ptr) * n);
    for (int i = 0; i < n; ++i)
    {
        ll val;
        scanf("%lld", &val);
        arr[i] = (node_ptr)malloc(sizeof(node));
        arr[i]->val = val;
        arr[i]->rank = i;
        to_arr[i] = arr[i];
    }

    qsort(arr, n, sizeof(node_ptr), comp);

    node_ptr head = (node_ptr)malloc(sizeof(node));
    head->prev = NULL;
    head->rank = -1;
    head->val = -1;
    node_ptr cur = head;
    for (int i = 0; i < n; ++i)
    {
        cur->next = arr[i];
        cur->next->prev = cur;
        cur = cur->next;
    }
    node_ptr tail = (node_ptr)malloc(sizeof(node));
    tail->prev = NULL;
```

```
tail->rank = -1;
tail->val = -1;
cur->next = tail;
cur->next->prev = cur;

node_ptr pr, ne;
ll prev_mwd = -1, next_mwd = -1;
for (int i = 0; i < n - 1; ++i)
{
    cur = to_arr[i];
    pr = cur->prev;
    ne = cur->next;
    prev_mwd = next_mwd = -1;
    while (pr->rank < cur->rank && NULL != pr->prev)
        pr = pr->prev;
    if (pr->rank != -1)
        prev_mwd = (pr->val > cur->val) ? pr->val - cur->val : cur->val - pr->val;

    while (ne->rank < cur->rank && NULL != ne->next)
        ne = ne->next;
    if (ne->rank != -1)
        next_mwd = (ne->val > cur->val) ? ne->val - cur->val : cur->val - ne->val;

    if (prev_mwd == -1)
        res[i] = next_mwd;
    else if (next_mwd == -1)
        res[i] = prev_mwd;
    else
        res[i] = (prev_mwd < next_mwd) ? prev_mwd : next_mwd;

    cur->prev->next = cur->next;
    cur->next->prev = cur->prev;
    node_ptr tmp = cur;
    cur = to_arr[i + 1];
    free(tmp);
}

for (int i = 0; i < n - 1; ++i)
{
    printf("%lld\n", res[i]);
}

cur = head;
while (cur)
{
    node_ptr tmp = cur;
    cur = cur->next;
    free(tmp);
}
```

```
    return 0;
}
```

B. Integer Editor

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct link_node *node_ptr;
typedef struct link_node
{
    char c;
    node_ptr prev;
    node_ptr next;
} node, *node_ptr;

char edits[100010];

int main()
{
    int Q;
    scanf("%d\n", &Q);
    scanf("%s", &edits);
    node_ptr eol = (node_ptr)malloc(sizeof(node));
    eol->c = '\n';
    eol->next = NULL;
    eol->prev = NULL;
    node_ptr cur = eol;
    for (int i = 0; i < Q;)
    {
        char edit = edits[i++];
        if (edit >= '0' && edit <= '9')
        {
            node_ptr tmp = (node_ptr)malloc(sizeof(node));
            tmp->c = edit;
            tmp->next = cur;
            tmp->prev = NULL;
            if (NULL != cur->prev)
            {
                cur->prev->next = tmp;
                tmp->prev = cur->prev;
            }
            cur->prev = tmp;
        }
        else
        {
            switch (edit)
            {
            case 'r':
            {
                char next_edit = edits[i++];
                if (NULL != cur->next)
                    cur->c = next_edit;
                else
                {
                    node_ptr tmp = (node_ptr)malloc(sizeof(node));
                tmp->c = next_edit;
                tmp->next = cur;
                tmp->prev = NULL;
                if (NULL != cur->prev)
                {
                    cur->prev->next = tmp;
                    tmp->prev = cur->prev;
                }
                cur->prev = tmp;
                cur = tmp;
            }
            break;
        }
        case 'I':
            while (NULL != cur->prev)
                cur = cur->prev;
            break;
        case 'L':
            if (NULL != cur->prev)
                cur = cur->prev;
            break;
        case 'R':
            if (NULL != cur->next)
                cur = cur->next;
            break;
        case 'd':
            if (NULL != cur->prev && NULL != cur->next)
            {
                cur->prev->next = cur->next;
                cur->next->prev = cur->prev;
                node_ptr tmp = cur;
                cur = cur->next;
                free(tmp);
            }
            else if (NULL != cur->next)
            {
                cur->next->prev = NULL;
                node_ptr tmp = cur;
                cur = cur->next;
                free(tmp);
            }
            break;
        default:
            break;
        }
    }
}
cur = eol;
int cnt = 0;
while (NULL != cur)
{
        edits[cnt] = cur->c;
        node_ptr tmp = cur;
        cur = cur->prev;
        free(tmp);
        ++cnt;
    }
    free(cur);
    while (cnt--)
    {
        printf("%c", edits[cnt]);
    }
    return 0;
}
```