Bài tập về nhà Stack and Queue

Bài 1:

```c
#include<stdio.h>
#include<string.h>

#define MAX 1001

typedef struct
{
    char a[MAX];
    int top;
} Stack;

void init(Stack *s){
    s->top = -1;
}

int isEmpty(Stack* s) {
    if(s->top == -1) {
        return 1;
    }else return 0;
}
int isFull(Stack* s) {
    if(s->top == MAX - 1) {
        return 1;
    }
    return 0;
}
void push(Stack* s, char value) {
    s->a[++s->top] = value;
}
char pop(Stack* s) {
    char value = s->a[s->top];
    --s->top;
    return value;
}
void displayStack(Stack* s) {
    printf("\nStack: ");
    for(int i = 0; i <= s->top; i++) {
        printf("%c", s->a[i]);
    }
    printf("\n");
```

```
}


int main() {
    char str1[1001];
    char str2[1001];
    scanf("%s", str1);
    for(int i = 0; i < strlen(str1); i++) {
        str2[i] = str1[i];
    }
    // Method 1:
    int i = 0;
    int j = strlen(str1) - 1;
    while(i < j) {
        char tmp = str1[i];
        str1[i] = str1[j];
        str1[j] = tmp;
        i++; j--;
    }
    printf("Method 1: %s\n",str1);

    Stack st;
    init(&st);

    for(int i = 0; i < strlen(str2); i++) {
        push(&st, str2[i]);
    }
    printf("Method 2: ");
    for(int i = 0; i < strlen(str2); i++) {
        printf("%c", pop(&st));
    }
    return 0;
}
```

Bài 2:

```
#include<stdio.h>
#include<string.h>
#define MAX 1001

typedef struct
{
    char a[MAX];
    int top;
} Stack;
```

```c
void init(Stack *s){
    s->top = -1;
}

int isEmpty(Stack* s) {
    if(s->top == -1) {
        return 1;
    }else return 0;
}
int isFull(Stack* s) {
    if(s->top == MAX - 1) {
        return 1;
    }
    return 0;
}
void push(Stack* s, char value) {
    s->a[++s->top] = value;
}
char pop(Stack* s) {
    char value = s->a[s->top];
    --s->top;
    return value;
}
char top(Stack *s) {
    return s->a[s->top];
}
int find(char c, char s[]) {
    for(int i = 0; i < strlen(s); i++) {
        if(c == s[i])
            return 1;
    }
    return 0;
}

void deleteArray(char a[], int *n, int index) {
    for(int i = index; i < *n - 1; i++) {
        a[i] = a[i + 1];
    }
    a[*n - 1] = '\0';
    (*n)--;
}

int main() {
    // Nhap du lieu ----------------------------------
```

```
    Stack st;
    init(&st);
    char str[1001] = "}{3+5*(4-1]}";
    // scanf("%s", str);
    int n = strlen(str);
    int error[1001] = {};
    int index_error = -1;
    char values[] = {')', '}', ']'};
    // truc quan hoa-----------------------------------
    for(int i = 0; i < n; i++) {
        printf("%2c ", str[i]);
    }
    printf("\n");
    for(int i = 0; i < n; i++) {
        printf("%2d ", i);
    }
    printf("\n");
    //-------------------------------------------------
    for(int i = 0; i < n; i++) {
        if(str[i] == '(' || str[i] == '[' || str[i] == '{') {
            push(&st, str[i]);
        }
        else if(find(str[i], values)) {
            if(isEmpty(&st)) {
                error[++index_error] = i;
                str[i] = ' ';
            }else {
                char c = str[i];
                if(c == ')') {
                    c -= 1;
                }else c -= 2;
                if(top(&st) != c) {
                    error[++index_error] = i;
                    str[i] = top(&st) == '('?top(&st) + 1:top(&st) + 2;
                    char tmp = pop(&st);
                }else {
                    char tmp = pop(&st);
                }
            }
        }else continue;
    }
    // Xu ly khoang cach----------------------------
    for(int i = 0; i < n; i++) {
        if(str[i] == ' ') {
            deleteArray(str, &n, i);
```

```
            i-=1;
        }
    }
    // Xuat ket qua---------------------------------
    if(index_error == -1) {
        printf("No error.\n");
    }else {
        printf("Errors at ");
        for(int i = 0; i <= index_error; i++) {
            printf("%d", error[i]);
            if(i < index_error) printf(", ");
        }
        printf("\n%s", str);
    }

    return 0;
}
```

Bài 3:

```
#include <stdio.h>
#include <string.h>
#define MAX 30

typedef struct
{
    char a[MAX];
    int top;
} Stack_c;

typedef struct
{
    int a[MAX];
    int top;
} Stack_i;

void init(Stack_c *s) { s->top = -1; }

void init1(Stack_i *s) { s->top = -1; }

int isEmpty(Stack_c *s)
{
    if (s->top == -1)
    {
        return 1;
```

```
    }
    else
        return 0;
}

int isEmptyi(Stack_i *s)
{
    if (s->top == -1)
    {
        return 1;
    }
    else
        return 0;
}

void pop(Stack_c *s)
{
    if (!isEmpty(s))
    {
        --s->top;
    }
    return;
}

void popi(Stack_i *s)
{
    if (!isEmptyi(s))
    {
        --s->top;
    }
    return;
}

void push(Stack_c *s, char c)
{
    s->top++;
    s->a[s->top] = c;
}

void pushi(Stack_i *s, int v)
{
    s->top++;
    s->a[s->top] = v;
}
```

```
char top(Stack_c *s) { return s->a[s->top]; }

char topi(Stack_i *s) { return s->a[s->top]; }

int getPrecedence(char op)
{
    if (op == '+' || op == '-')
        return 1;
    if (op == '*' || op == '/')
        return 2;
    return 0;
}

int is_digit(char c)
{
    if (c >= '0' && c <= '9')
        return 1;
    return 0;
}

void solve(char s[])
{
    Stack_c st1;
    init(&st1);
    int n = strlen(s);
    char hauto[MAX] = "";
    int idx = 0; // chi so cua hau to
    for (int i = 0; i < n; i++)
    {
        if (is_digit(s[i]))
        {
            hauto[idx++] = s[i];
        }
        else
        {
            while (!isEmpty(&st1) && getPrecedence(top(&st1)) >=
getPrecedence(s[i]))
            {
                hauto[idx++] = top(&st1);
                pop(&st1);
            }
            push(&st1, s[i]);
        }
    }
    while (!isEmpty(&st1))
```

```
    {
        hauto[idx++] = top(&st1);
        pop(&st1);
    }

    printf("%s\n", hauto);
    // tinh toan -------------
    Stack_i st2;
    init1(&st2);
    for (int i = 0; i < strlen(hauto); i++)
    {
        if (is_digit(hauto[i]))
        {
            pushi(&st2, hauto[i] - '0');
        }
        else
        {
            int t2 = topi(&st2);
            popi(&st2);
            int t1 = topi(&st2);
            popi(&st2);
            if (hauto[i] == '+')
            {
                pushi(&st2, t1 + t2);
            }
            else if (hauto[i] == '-')
            {
                pushi(&st2, t1 - t2);
            }
            else if (hauto[i] == '*')
            {
                pushi(&st2, t1 * t2);
            }
            else
            {
                if (t2 != 0)
                {
                    pushi(&st2, t1 / t2);
                }
                else
                {
                    printf("Lỗi chia cho 0.\n");
                    return;
                }
            }
```

```
        }
    }
    printf("%d", topi(&st2));
}

int main()
{
    char s[] = "2+3*4-5";
    solve(s);
    return 0;
}
```

Bài 4:

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

#define MAX 7
#define null INT_MIN

typedef struct {
    int head, tail;
    int a[MAX];
} Queue;

void init(Queue* q) {
    q->head = 0;
    q->tail = -1;
    for(int i = 0; i < MAX; i++) {
        q->a[i] = null;
    }
}

int isEmpty(Queue *q) {
    if(q->head == 0 && q->tail == -1) {
        return 1;
    }
    return 0;

}

int isFull(Queue *q) {
    if(q->head < q->tail) {
        if(q->tail == MAX - 1) return 1;
```

```
        return 0;
    }
    else{
        if(!isEmpty(q)) {
            if(q->tail + 1 == q->head) {
                return 1;
            }
            else return 0;
        }else {
            return 0;
        }
    }
}

void put(Queue* q, int value) {
    if(isFull(q)) {
        printf("\nQueue is full!\n");
        return;
    }
    if(q->tail < MAX - 1) { // 5
        q->tail += 1;
        q->a[q->tail] = value;
    }else {
        q->tail = -1;
        if(q->tail + 1 < q->head) {
            q->tail += 1;
            q->a[q->tail] = value;
        }else {
            q->tail = MAX - 1;
            // printf("Queue is full!\n");
        }
    }
}

int get(Queue* q) {
    if(isEmpty(q)) {
        return null;
    }
    int index = q->head;
    int value;
    if(q->head == q->tail) {
        // bao dong
        q->tail = -1;
        q->head = 0;
        value = q->a[index];
```

```
        q->a[index] = null;
    }else if(index == MAX - 1){
        value = q->a[index];
        q->a[index] = null; // danh dau phan tu da bi xoa
        q->head = 0;
    }else {
        value = q->a[index];
        q->a[index] = null;
        q->head += 1;
    }

    return value;
}

void displayQueue(Queue*q) {
    if(isEmpty(q)) {
        printf("Empty Queue!\n");
        return;
    }
    printf("\nQueue: ");
    if(q->head <= q->tail) {
        for(int i = q->head; i <= q->tail; i++) {
            printf("%3d ", q->a[i]);
        }
    }else {
        for(int i = q->tail; i < MAX; i++) {
            printf("%3d ", q->a[i]);
        }
        for(int i = 0; i <= q->head; i++) {
            printf("%3d ", q->a[i]);
        }
    }
    printf("\n");
}
void displayQueue1(Queue *q) {
    printf("\nQueue Status: ");
    for(int i = 0; i < MAX; i++) {
        if(q->a[i] != null) {
            printf("%4d ", q->a[i]);
        }else printf("null ");
    }
    printf("\n");
}
void deleteArray(int a[], int n, int index) {
    int len = n - 1;
```

```
    for(int i = index; i <= len; i++) {
        a[i] = a[i + 1];
    }
    a[n] = null;
}
void cancelRegistration(Queue *q, int value) {
    if(isEmpty(q)) {
        printf("Queue is empty!\n");
        return;
    }
    // Tim index can xoa
    int index = -1;
    for(int i = 0; i < MAX; i++) {
        if(q->a[i] == value){
            index = i;
            break;;
        }
    }
    if(index == -1) {
        printf("Khong tim thay\n");
        return;
    }else {
        printf("\nXoa index = %d, value = %d\n", index, value);
    }
    // Xu ly---------------
    if(q->head <= q->tail) {
        deleteArray(q->a, q->tail, index);
        q->tail--;
    }else {
        if(index >= q->head) {
            for(int i = index; i < MAX - 1; i++) {
                q->a[i] = q->a[i + 1];
            }
            q->a[MAX - 1] = q->a[0];
            for(int i = 0; i < q->tail; i++) {
                q->a[i] = q->a[i + 1];
            }
            q->a[q->tail] = null;
        }else if (index <= q->tail){
            for(int i = index; i < q->tail; i++) {
                q->a[i] = q->a[i + 1];
            }
            q->a[q->tail] = null;
        }
        q->tail--;
```

```
    }
    if(q->tail + 1 == q->head) {
        q->head = 0; q->tail = -1; // Neu queue da rong thi reset lai
    }
}

int main() {
    // pass //
    Queue q;
    init(&q);
    int id[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

    /////LUU Y : ham displayQueue() in ra theo thu tu tu head den tail, ham
displayQueue1() in ra trang thai cua mang
    // Put 7 thang vao ---------------
    for(int i = 0; i < sizeof id / sizeof (int); i++) {
        put(&q, id[i]);
        // displayQueue1(&q);
    }
    printf("----------------------------\n");
    printf("\nSau khi put\n");
    displayQueue1(&q);
    // displayQueue1(&q);
    // Get 3 thang ra ----------------------
    for(int i = 0; i < 3; i++) {
        get(&q);
        // displayQueue1(&q);
    }
    printf("----------------------------\n");
    printf("\nSau khi get\n");
    // Xoa 4 thang ------------------------
    displayQueue1(&q);
    int xoa[] = {7, 6, 5, 4};
    for(int i = 0; i < sizeof xoa / sizeof (int); i++) {
        cancelRegistration(&q, xoa[i]);
        displayQueue(&q);
        displayQueue1(&q);
        printf("----------------------------\n");
    }
    // Sau khi xoa hang doi bi rong~
    return 0;
}
```

Bài 5:

```c
#include <stdio.h>
#include<stdlib.h>
#define MAX 30
#define null INT_MIN
typedef struct {
    int head, tail;
    int a[MAX];
} Queue;


void init(Queue* q) {
    q->head = 0;
    q->tail = -1;
    for(int i = 0; i < MAX; i++) {
        q->a[i] = null;
    }
}

int isEmpty(Queue *q) {
    if(q->head == 0 && q->tail == -1) {
        return 1;
    }
    return 0;

}

int isFull(Queue *q) {
    int numval = 0;
    if(q->head < q->tail) {
        numval += q->tail - q->head + 1;
        if(numval == MAX)
            return 1;
        else return 0;
    }
    else{
        if(!isEmpty(q)) {
            if(q->tail + 1 == q->head) {
                return 1;
            }
            else return 0;
        }else {
            return 0;
```

```
        }
    }
}

void put(Queue* q, int value) {
    if(q->tail < MAX - 1) { // 5
        q->tail += 1;
        q->a[q->tail] = value;
    }else {
        q->tail = -1;
        if(q->tail + 1 < q->head) {
            q->tail += 1;
            q->a[q->tail] = value;
        }else {
            q->tail = MAX - 1;
            // printf("Queue is full!\n");
        }
    }
}

int get(Queue* q) {
    if(isEmpty(q)) {
        return null;
    }
    int index = q->head;
    int value;
    if(q->head == q->tail) {
        // bao dong
        q->tail = -1;
        q->head = 0;
        value = q->a[index];
        q->a[index] = null;
    }else if(index == MAX - 1){
        value = q->a[index];
        q->a[index] = null; // danh dau phan tu da bi xoa
        q->head = 0;
    }else {
        value = q->a[index];
        q->a[index] = null;
        q->head += 1;
    }

    return value;
}
```

```c
void displayQueue(Queue*q) {
    printf("\nQueue: ");
    for(int i = 0; i < MAX; i++) {
        if(q->a[i] != INT_MIN) {
            printf("%4d ", q->a[i]);
        }else printf("null ");
    }
    printf("\n");
}


void swap(int *a, int *b) {
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

void selectionSort(int a[], int n) {
    for(int i = 0; i < n - 1; i++) {
        int min_idx = i;
        for(int j = i + 1; j < n; j++) {
            if(a[j] < a[min_idx])
                min_idx = j;
        }
        swap(&a[i], &a[min_idx]);
    }
}
void deleteArray(int a[], int n, int index) {
    int len = n - 1;
    for(int i = index; i <= len; i++) {
        a[i] = a[i + 1];
    }
    a[n] = null;
}
void cancelRegistration(Queue *q, int value) {
    if(isEmpty(q)) {
        // printf("Queue is empty!\n");
        return;
    }
    // Tim index can xoa
    int index = value;
    // for(int i = 0; i < MAX; i++) {
    //     if(q->a[i] == value){
    //         index = i;
    //         break;;
```

```
    //     }
    // }
    if(index == -1) {
        // printf("Khong tim thay\n");
        return;
    }else {
        // printf("\nXoa index = %d, value = %d\n", index, value);
    }
    // Xu ly---------------
    if(q->head <= q->tail) {
        deleteArray(q->a, q->tail, index);
        q->tail--;
    }else {
        if(index >= q->head) {
            for(int i = index; i < MAX - 1; i++) {
                q->a[i] = q->a[i + 1];
            }
            q->a[MAX - 1] = q->a[0];
            for(int i = 0; i < q->tail; i++) {
                q->a[i] = q->a[i + 1];
            }
            q->a[q->tail] = null;
        }else if (index <= q->tail){
            for(int i = index; i < q->tail; i++) {
                q->a[i] = q->a[i + 1];
            }
            q->a[q->tail] = null;
        }
        q->tail--;
    }
    if(q->tail + 1 == q->head) {
        q->head = 0; q->tail = -1; // Neu queue da rong thi reset lai
    }
}

int solve(Queue *q) {
    if(isEmpty(q)) {
        return 0;
    }
    int count = 0;
    for(int i = q->head; i <= q->tail && !isEmpty(q); i++) {
        if(q->a[i] == 1) {
            cancelRegistration(q, i);
            i-=1;
            count += 1;
```

```
        }else q->a[i]--;
    }
    return count;
}

int main() {
    Queue q;
    init(&q);
    int k, n;
    int num[MAX];
    scanf("%d%d", &k, &n);
    for(int i = 0; i < n; i++) scanf("%d", &num[i]);
    // selectionSort(num, n);
    for(int i = 0; i < k; i++) {
        put(&q, num[i]);
    }
    int totaltime = 0;
    int totalnum = k;
    displayQueue(&q);
    while(!isEmpty(&q)){
        totaltime++;
        int cnt = solve(&q);
        if(totalnum < n) {
            for(int i = totalnum; i < totalnum + cnt; i++) put(&q, num[i]);
        }
        totalnum += cnt;
    displayQueue(&q);
    }
    printf("total_time = %d\n", totaltime);


}
```