第三四章作业

选择题

1. C
2. C
3. C
4. A
5. B
6. D
7. D
8. C

简答题

1. 把栈中元素依次出栈，再依次入栈，就可以得到逆序的栈。
2. 把队列元素依次出队压入栈中，再依次出栈入队列，就可以得到逆序的队列。
3. 为了解决表达式 `3-2*8/4+3^2`

表达式读取进度	操作数栈	运算符栈
3	3	(
3-	3	(, -
3-2	3, 2	(, -
3-2*	3, 2	(, -, *
3-2*8	3, 2, 8	(, -, *
3-2*8/	3, 16	(, -
3-2*8/4	3, 16, 4	(, -, /

表达式读取进度	操作数栈	运算符栈
3-2*8/4	3, 4	(, -
3-2*8/4	-1	(,
3-2*8/4+	-1	(, +
3-2*8/4+3	-1, 3	(, +
3-2*8/4+3^	-1, 3	(, +, ^
3-2*8/4+3^2	-1, 3, 2	(, +, ^
3-2*8/4+3^2	-1, 9	(, +
3-2*8/4+3^2	8	(,

答案是 8。

4. 首先，让我们计算模式串的 `next` 函数值。
   主串 s = ADBADABBAABADADABBADADA
   模式串： ADABBADADA

next数组：0 1 1 2 1 1 2 3 4 3
nextval数组：0 1 0 2 1 0 1 0 4 0

则匹配过程如下：

主串： ADBADABBAABADADABBADADA
模式串： ADABBADADA

1. 首先，比较主串和模式串的第一个字符，它们匹配，所以移动到下一个字符。
2. 接下来，比较主串和模式串的第二个字符，它们也匹配，所以再次移动到下一个字符。
3. 现在，比较主串的第三个字符和模式串的第三个字符，它们不匹配。根据 `nextval` 数组，模式串的指针移动到位置0，主串的指针实际上+1。

主串： ADBADABBAABADADABBADADA
模式串： ADABBADADA

4. 然后，比较主串的第四个字符和模式串的第一个字符，它们匹配，所以移动到下一个字符。
5. 直到比较主串的第十个字符和模式串的第七个字符，它们不匹配。根据 `nextval` 数组，模式串的指针移动到位置1，主串的指针保持不变。

主串：　ADBADABBAABADABBADADA
模式串：　　ADABBADADA

6. 此过程将继续，直到主串的指针到达位置12。

主串：　ADBADABBAABADABBADADA
模式串：　　ADABBADADA

7. 此时，从主串的位置12和模式串的位置1开始，所有后续的字符都匹配，直到模式串的最后一个字符。

这样，就找到了匹配的子串。

算法设计题

1. 代码如下：

```c
typedef struct {
    ElemType data[MAXSIZE];
    int top1;  // 栈1的栈顶指针
    int top2;  // 栈2的栈顶指针
} DoubleStack;

// 初始化栈
void InitStack(DoubleStack *tws) {
    tws->top1 = -1;        // 第一个栈的栈顶初始化为-1
    tws->top2 = MAXSIZE;  // 第二个栈的栈顶初始化为MAXSIZE
}

// 入栈操作
bool push(DoubleStack *tws, int i, ElemType e) {
    if (tws->top1 + 1 == tws->top2) {  // 判断栈是否已满
        printf("Stack Overflow!\n");
        return false;
    }
    if (i == 0) {
        tws->data[++tws->top1] = e;
    } else if (i == 1) {
        tws->data[--tws->top2] = e;
    }
    return true;
}

// 出栈操作
bool pop(DoubleStack *tws, int i, ElemType *e) {
    if (i == 0) {
        if (tws->top1 == -1) {  // 判断栈1是否为空
            printf("Stack1 Underflow!\n");
            return false;
        }
        *e = tws->data[tws->top1--];
    } else if (i == 1) {
        if (tws->top2 == MAXSIZE) {  // 判断栈2是否为空
            printf("Stack2 Underflow!\n");
            return false;
        }
        *e = tws->data[tws->top2++];
    }
```

```
    return true;
}
```

2. 代码如下：

```c
typedef struct Node {
    ElementType data;
    struct Node* next;
} Node;

typedef struct {
    Node* rear;  // 只指向队尾元素结点
} Queue;

Queue* initializeQueue() {
    Queue* q = (Queue*)malloc(sizeof(Queue));
    if (!q) {
        exit(1);  // 分配内存失败
    }
    q->rear = (Node*)malloc(sizeof(Node));  // 创建头结点
    if (!q->rear) {
        exit(1);  // 分配内存失败
    }
    q->rear->next = q->rear;  // 循环指向自己
    return q;
}

void enqueue(Queue* q, ElementType e) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        exit(1);  // 分配内存失败
    }
    newNode->data = e;
    newNode->next = q->rear->next;  // 新结点指向头结点
    q->rear->next = newNode;        // 队尾的下一个结点指向新结点
    q->rear = newNode;              // 更新队尾指针
}

ElementType dequeue(Queue* q) {
    if (q->rear == q->rear->next) {
        exit(1);  // 队列为空，出队失败
    }
    Node* front = q->rear->next->next;  // 第一个元素结点
    ElementType e = front->data;
    q->rear->next->next = front->next;  // 头结点的下一个结点指向第一个元素结点的下一个结点
    free(front);                        // 释放第一个元素结点
```

```
    return e;
}
```

3. 代码如下：

```
bool isPalindrome(char str[]) {
    int start = 0;            // 开始位置
    int end = strlen(str) - 1;  // 结束位置

    while(start < end) {
        if(str[start] != str[end]) {
            return false;  // 如果不相等，则不是回文
        }
        start++;
        end--;
    }

    return true;  // 所有字符都相等，所以是回文
}
```