第 1 题代码如下：

```
#include <stdio.h>
#include <stdlib.h>
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define OVERFLOW -2
#define STACK_INIT_SIZE 100          //存储空间初始分配量
#define STACKINCREMENT 10            //存储空间分配增量
typedef int Status;
typedef int SElemType;
typedef struct
{
    SElemType *base;      //在栈构造之前和销毁之后，base 的值为 NULL
    SElemType *top;       //栈顶指针
    int stacksize;        //当前已分配的存储空间，以元素为单位
}SqStack;

Status InitStack(SqStack &S){
    //构造一个空栈 S
    S.base = (SElemType * )malloc(STACK_INIT_SIZE * sizeof(SElemType));
    if (!S.base) exit(OVERFLOW);       //存储分配失败
    S.top = S.base;
    S.stacksize = STACK_INIT_SIZE;
    return OK;
}//InitStack

Status StackEmpty(SqStack &S){
    if _____________
        return TRUE;
    else
        return FALSE;
}//StackEmpty

Status GetTop(SqStack &S, SElemType &e){
    //若栈不空，则用 e 返回 S 的栈顶元素，并返回 OK；否则返回 ERROR
    if (S.top ==S.base) return ERROR;
    _____________
    return OK;
}//GetTop

Status Push(SqStack &S, SElemType e){
    //插入元素 e 为新的栈顶元素
    if (S.top - S.base >= S.stacksize){ //栈满，追加存储空间
        S.base = (SElemType * )realloc(S.base,(S.stacksize + STACKINCREMENT) * sizeof(SElemType));
        if (!S.base) exit(OVERFLOW);     //存储分配失败
        S.top = S.base + S.stacksize;
```

```
        S.stacksize += STACKINCREMENT;
    }
    _____
    return OK;
}//Push

Status Pop(SqStack &S, SElemType &e){
    //若栈不空，则删除 S 的栈顶元素，用 e 返回其值，并返回 OK；否则返回 ERROR
    if (S.top ==S.base) return ERROR;
    _____
    return OK;
}//Pop

Status Stackoutput(SqStack &S){
    //若栈不空，则从栈顶到栈底依次输出数据元素，并返回 OK；否则返回 ERROR
    SElemType *p;
    if (S.top ==S.base) return ERROR;
    _____
    while _____
        _____
    _____
    return OK;
}//StackTraverse

Status StackTraverse(SqStack &S){
    //若栈不空，则从栈底到栈顶依次输出数据元素，并返回 OK；否则返回 ERROR
    SElemType *p;
    if (S.top ==S.base) return ERROR;
    _____
    while _____
        _____
    _____
    return OK;
}//StackTraverse

void main()
{
    int i,n,k,h,a,b;
    SqStack S;
    printf("创建一个空栈！\n");
    InitStack(S);
    printf("判断栈是否为空！\n");
    printf("StackEmpty(S)=%d\n",StackEmpty(S));
    printf("创建栈的元素个数：\n");
    scanf("%d", &n);
    printf("输入%d 个入栈元素的值：\n",n);
    for(i=0;i<n;i++)
    {
```

```
        scanf("%d", &k);
        Push(S, k);
    }
    printf("逆序输出顺序栈元素值：\n");
    Stackoutput(S);
    printf("输出顺序栈元素值：\n");
    StackTraverse(S);
    printf("输入入栈元素值：");
    scanf("%d", &h);
    Push(S, h);
    printf("输出入栈后的顺序栈元素值：\n");
    StackTraverse(S);
    Pop(S, a);
    printf("输出第 1 个出栈元素值：%d\n", a);
    Pop(S, a);
    printf("输出第 2 个出栈元素值：%d\n", a);
    printf("输出两次出栈后顺序栈元素值：");
    StackTraverse(S);
    GetTop(S, b);
    printf("输出栈顶元素值：%d\n",b);
}
```

第 2 题部分代码如下：

```
typedef char SElemType;
typedef struct
{
    SElemType *base;      //在栈构造之前和销毁之后，base 的值为 NULL
    SElemType *top;       //栈顶指针
    int stacksize;        //当前已分配的存储空间，以元素为单位
}SqStack;
Status Correct(SElemType str[]){
    //使用运算符栈 S，当遇到'('、'['时进栈，遇到')'、']'出栈并判断出栈元素是否为相应的符号，
    //若是则继续下一个，否则算法结束
    SqStack S;
    InitStack(S);         // 构造空栈
    int i, state=1;
    SElemType e;
    for(i=0;str[i]!='\0';i++)
    {
        switch(str[i])
        {
        case '(':  ___________  ________
        case '[':  ___________  ________
        case ')':
            ______________
            ______________
            ______________
```

```
            case ']':
                ____________
                ____________
                ____________
            }
            if (!state) break;
        }
        if ( ____________ && state ==1)
            return OK;
        else
            return ERROR;
}//Correct

void main()
{
        SElemType str[100];
        printf("请输入带括号的表达式：\n");
        scanf("%s", str);
        if (Correct(str) ==OK)
            printf("括号匹配正确！\n");
        else
            printf("括号匹配不正确！\n");
}
```