声通实验室 2021 级软件第一次考核

1. 考核对象：2021 级声通实验室软件组成员（数据结构与算法）
2. 考核时间：2022.09.25-2022.10.07
3. 考核说明： 本次考核为数据结构与算法，共 1 个大题（内含选做题）+1 个附加题，满分 100（+100）分。
4. 提交说明：
   a 提交方式：将程序以.c 文件格式，连同需要提交的图片，共同压缩为一个.zip 或.rar 文件发送给管理员（云隐 1065415674），命名格式：年级专业姓名
   b 截止时间：2022.10.07 中午 12：00
5. 题目有一定难度，如果过程中有不懂的，大可以在群里问，考核的目的在于督促学习，而不是把大家难倒。提交后 24 小时内在 https://docs.qq.com/sheet/DTkp5TWxzV3RGV2lw?tab=BB08J2 在线表格中可以看到成绩及错误情况

1. 队列是一种简单的数据结构，它的思想是先进先出，实现方式有数组与链表等。请学习完相关概念后，使用 C 语言，完成以下要求：

(1) 使用数组作为基本实现方式，完成以下函数：（50 分）

给定条件：

```
int q[100];

int front = 0;

int rear = 0;
```

① int pop()（10 分）

1) 参数：无

2) 返回值：int：出队的元素值

3) 功能：将队列里的第一个元素出队，并返回出队的元素值

② int push(int data)（10 分）

1) 参数：int 型的元素数据

2) 返回值：int：1 代表入队成功，0 代表队列已满

3) 功能：将一个元素入队

③ int isempty()（10 分）

1) 参数：无

2) 返回值：int：1 代表队列为空，0 代表队列不为空

3) 功能：检测队列是否为空

④ int front()（10 分）

1) 参数：无

2) 返回值：队列的第一个元素值

3) 功能：返回队列的第一个元素，如果队列为空，打印“warning：queue isempty!”

⑤ int size()（10 分）

1) 参数：无

2) 返回值：队列里的元素个数

3) 功能：返回队列里的元素个数

(2) （C++选做）请使用 C++中的类模板，查阅 STL 库中队列类的函数，完成如下队列的实现：（50 分）

template <typename T>

```
class Myqueue
{
private:
    int front;     //头元素下标
    int rear;      //尾元素下标
    int maxSize; //队列总空间大小
    T *array;

public:
    Myqueue(int size = 10); //构造函数
    ~Myqueue(); //析构函数
    T back();
    bool empty();
    T frontVal();
    void pop();
    void push(T value);
    int size();
};
```

（3）请使用你制作的队列函数测试以下任务：

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

```
int q[100];
int head = 0;
int rear = 0;
//请在此处插入你第（1）问的函数
int main()
{
    int a[] = { 1,2,9,6,5,4,7,12,5,0,21 };
    for (int i = 0; i < sizeof(a) / sizeof(a[0]); i++)
    {
        push(a[i]);
        printf("%d 已入队", a[i]);
    }
    if (front() == 1)
        printf("front 正确");
    while (!isempty())
    {
        printf("%d 已出队\n", pop());
        printf("剩余%d 个元素", size());
    }
    return 0;
```

}

//请将运行结果截图放入压缩包内

2. （附加题）算法的基础是数学，如果想知其然，至少从会算一些简单的时间复杂度开始。请使用大 O 表示法，写出下列算法（或表达式）的时间复杂度。（如果有推导过程，可以写在纸上放入压缩包内）（50 分）

(1) F(N) = F(N-1) + O(1)　（10 分）

(2) F(N) = 2F(N/2) + O(N)　（15 分）

(3) 冒泡排序：（15 分）

```
void bubble_sort(int arr[], int len) {
    int i, j, temp;
    for (i = 0; i < len - 1; i++)
        for (j = 0; j < len - 1 - i; j++)
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
}
```

（4）最大子序列和的一种算法：（20 分）

```
int MaxSubSum(const int A[], int left, int right)
{
    if (left == right)
        if (A[left] > 0)
            return A[left];
        else
            return 0;
    int centre = (left + right) / 2;
    int leftMax = MaxSubSum(A, left, centre);
    int rightMax = MaxSubSum(A, centre + 1, right);

    int left_temp_Max = 0, left_C_Max = 0;
    int right_temp_Max = 0, right_C_Max = 0;
    for (int i = centre; i >= left; i--)
    {
        left_temp_Max += A[i];
        if (left_temp_Max > left_C_Max)
            left_C_Max = left_temp_Max;
    }
```

```
    for (int i = centre + 1; i <= right; i++)
    {
        right_temp_Max += A[i];
        if (right_temp_Max > right_C_Max)
            right_C_Max = right_temp_Max;
    }
    return Max(right_C_Max + left_C_Max, leftMax, rightMax);
}
```