1. 填空题

1. 堆栈和队列的共同之处在于它们具有共同的\_\_\_\_\_\_\_

A. 逻辑特性 B. 物理特性 C. 运算方法 D. 元素类型

2. 若某堆栈初始为空，PUSH与POP分别表示对堆栈进行一次进栈和出栈操作，那么对于进栈序列，a,b,c,d,e,经过PUSH,PUSH,POP,PUSH,POP,PUSH,PUSH以后，得到的出栈序列是\_\_\_\_\_\_\_

A. b,a B. b,c C. b,d D. b,e

3. 若5个元素的出栈序列是1,2,3,4,5，则进栈序列可能是\_\_\_\_\_\_\_\_

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

4. 某队列初始为空，若它的输入序列为a,b,c,d,他的输出序列应为\_\_\_\_\_\_

A. a,b,c,d B. d,c,b,a C. a,c,b,d D. d,a,c,b

5. 当四个元素的进栈序列定了以后，有该4个元素组成的可能的出栈序列应该有\_\_\_\_

A. 24种 B. 17种 C. 16种 D. 14种

6. 设n个元素的进栈序列为出栈序列为，若,则的值\_\_\_\_\_\_

A. 为I B. 为n-I C. 为n-i+1 D. 有多种可能

7. 设n个元素的进栈序列为出栈序列为，若,则的值\_\_\_\_\_\_

A. 为I B. 为n-I C. 为n-i+1 D. 有多种可能

8. 若堆栈采用顺序存储结构，正常情况下，往堆栈中插入一个元素，栈顶指针top的变化是\_\_\_\_\_\_\_\_

A. 不变 B. top=0 C. --top D. ++top

9. 若堆栈采用顺序存储结构，正常情况下，删除堆栈中一个元素，栈顶指针top的变化是\_\_\_\_\_\_\_\_

A. 不变 B. top=0 C. --top D. ++top

10. 若队列采用顺序存储结构，元素的排列顺序是\_\_\_\_\_\_

A. 与元素值的大小有关 B.由元素进入队列的先后顺序决定

C. 与队头指针与队尾指针的取值有关 D. 与作为顺序存储结构的数组的大小有关

11. “链接队列”这一概念不涉及\_\_\_\_\_\_

A. 数据的存储结构 B. 数据的逻辑结构

C. 对数据进行的操作 D. 链表的种类

12. 若堆栈采用链式存储结构，栈顶指针为top,向堆栈插入一个由p指向的新结点的过程是依次执行: \_\_\_\_\_\_\_\_\_,top=p

A. p=pop B. top=p C. p->link=top D. top->link=p

13. 若非空堆栈采用链式存储结构，栈顶指针为top，删除堆栈一个元素的过程是依次执行：p=top;,\_\_\_\_\_,free(p);

A. top=p B. top=p->link C. p=top->link D. p=p->link

14. 若队列采用链式存储结构，队头元素指针与队尾元素指针分别为front和rear,向队列中插入一个由p指向的新结点的过程是依次执行：\_\_\_\_\_\_\_\_,rear=p;

A. rear=p; B. front=p; C. rear->link=p; D. front->link=p;

15. 若非空队列采用链式存储结构，队头元素指针与队尾元素指针分别为front和rear，删除队列的一个元素的过程是依次执行：p=front;,\_\_\_\_,free(p);

A. rear=p; B. rear=p->link C. front=rear->link D. front=p->Link

16. 在循环队列中，若front和rear分别表示队头元素与队尾元素的位置，则判断循环队列队空的条件是\_\_\_\_\_\_\_

A. front=rear+1 B. rear = p->link

C. front=rear D. front = p->link

17. 描述某循环队列的数组为QUEUE,当循环队列满时，队列中有\_\_\_\_个元素

A. M B. M-1 C. M+1 D. M+2

18. 在解决计算机主机与打印机之间速度不匹配的问题时，通常设置一个打印数据缓冲区，主机将要输出的数据依次写入该缓冲区，而打印机则依次从该缓冲区中取出数据打印，该缓冲区应该是个\_\_\_\_\_\_\_结构

A. 线性表 B. 数组 C. 堆栈 D. 队列

19. 设计一个递归问题的非递归算法通常需要设置\_\_\_\_\_\_结构

A. 线性表 B. 数组 C. 堆栈 D. 队列

20. 中缀表达式A-(B+C/D)XE的后缀形式是\_\_\_\_\_\_\_

A. ABC+D/XE- B. ABCD/+EX-

C. AB-C+D/EX D. ABC-+D/EX

二、填空题

1. 堆栈和队列的逻辑结构都是\_\_\_\_\_\_结构

2. 堆栈的插入和删除操作都是在\_\_\_\_\_\_\_位置进行，而队列的插入操作在\_\_\_\_\_进行，删除操作在\_\_\_\_\_\_\_进行

3. 对某堆栈执行删除操作时，只有在\_\_\_\_\_\_\_情况下，才会将栈底元素删除

4. 在具体的程序设计过程中，堆栈的顺序存储结构一般是利用一个\_\_\_\_\_\_描述的，同时还要定义一个整型变量来\_\_\_\_\_\_\_

5. 堆栈采用顺序存储结构，在未溢出的情况下往堆栈中插入一个元素，首先\_\_\_\_\_\_然后\_\_\_\_\_\_\_\_

6. 队列采用顺序存储结构，未溢出时插入一个元素首先\_\_\_\_\_\_，再\_\_\_\_\_

7. 当堆栈的最大长度难以估计时，堆栈最好采用\_\_\_\_\_\_存储结构

8. 递归算法都可以通过设置\_\_\_\_\_\_机制改写成等价的非递归算法

9. 中缀形式的算术表达式A+(B-C)/DXE的后缀形式是\_\_\_\_\_\_\_

10. 后缀形式的算术表达式ABCD/-EX+的中缀形式为\_\_\_\_\_\_\_\_\_

三、解答题

1. 已知堆栈采用链式存储结构，初始时为空，请画出a,b,c,d4个元素依次进栈以后该堆栈的状态，然后再画出此时的那个栈顶元素出栈后堆栈的状态

2. 若按从左到右的顺序依次读入已知队列(a,b,c,d,e,f,g)中的元素，然后结合堆栈操作，能得到下列序列中的哪些序列（每个元素进栈一次，下列序列表示出栈的顺序）

3. 设有编号为1,2,3,4 的4辆车，顺序进入一个栈式结构的站台，请写出这4辆车开出车站的所有可能顺序

4. 若n个元素的进栈次序为，则有多少中可能的输出序列

5. 设为个堆栈共享，若各栈栈顶指针为top[0…n-1]，请分别指出各个栈顶元素的位置，若栈顶指针为bot[0…n]，请分别指出各个栈底元素的位置，初始时有：

(I=1,2,…n)

其中ROUND()为四舍五入取整函数，请写一算法，该算法向任意指定的第i个堆栈插入一个新的元素x,仅当M个空间全部占用时才产生溢出，并报告相应信息

6. 设中缀表达式E存放于字符数组中，并以’@’作为结束标识。请写出判断一个中缀表达式E中左右圆括号是否配对的算法。

7. 写出将中缀表达式#(a+b)/c-d#变换为后缀表达式的过程中，每读到一个单词时堆栈的状态

8. 请利用一链接堆栈编写一非递归算法，对于给定的十进制整数num，打印出对应的r进制整数()

9. 已知n为大于等于零的整数，请写出利用堆栈计算下列递归函数f(n)的非递归算法

10. 已知Ackerman函数的定义如下：

(1). 写出相应的递归算法

(2). 利用堆栈写出非递归算法

(3). 根据非递归算法，求出ACK(2,1)的值

11. 已知求两个正整数m和n的最大公约数的过程可以表达为如下递归函数

请写出求该递归函数的非递归算法，（n MOD n表示求m除以n的余数）

12. 假设以数组Q[0..M-1]存放循环队列的元素，同时设置变量rear与qlen分别指示循环队列中队尾元素的位置和队列中元素的个数，请给出此循环队列的队满条件，并写出相应的进队与出队算法（在出队算法中要求返回队头元素）

13. 一个双向队列DQ是限定在两端end1和end2都可以进行插入和删除操作的线性表，队空的条件是end1=end2.若该双向队列采用顺序存储结构，试根据下列要求定义双向队列的结构，并给出在指定端endi(i=1,2)进行插入和删除操作的算法。要求：

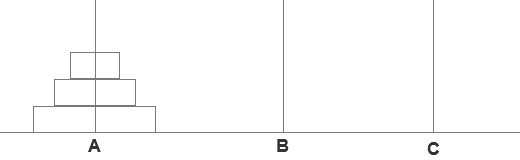
1). 当双向队列满时，最多只能有一个数据元素的空间是空的

2). 在进行两端的插入和删除操作时，双向队列中的其他数据元素一律不动

14. 请写出利用堆栈解决迷宫问题的递归算法

15. 梵塔问题是这样的：一个底盘上有3根竖着的真，初始时A针穿着n张盘片，现要求将这n张盘片移到c针上，并且任何时刻不得将大盘放到小盘上，而且每一次只允许移动一张盘片，写一算法，打印出正确的操作步骤

提示：将n张盘片由A依次移到C针上，B作为辅助针，当n=1时，可以直接完成，否则将顶上的n-1张盘片移到B针上，用C针作为辅助针，然后移第n张盘片，最后将B上的n-1张盘片移到C针上，并用A针作为辅助针



答案：

一、填空题

1. A

2. B

3. D

4. A

5. D

6. C

7. C

8. D

9. C

10. B

11. B

12. C

13. B

14. C

15. D

16. C

17. B

18. D

19. C

20. B

二、 填空题

1. 线性结构

2. 栈顶位置

3. 栈中只有一个元素的

4. 数组 给出栈顶元素的位置

5. 将栈顶指针后移一个位置 将被插入元素放在修改后的栈顶指针所指的位置

6. 将队尾指针后移一个位置 将被插入元素放在修改后的队尾指针所指出的位置

7. 链式

8. 堆栈

9. ABC-D/EX+

10. A+(B-C/D)XE

三、填空题

1. 略

2. 可以得到1和4

3. 这四辆列车开出车站的所有可能的顺序共有14种情况，分别为

1,2,3,4 1,2,3,4 1,3,2,4 1,3,4,2 2,1,3,4 1,4,3,2 2,1,4,3

2,3,1,4 2,3,4,1 2,4,3,1 3,2,1,4 3,2,4,1 3,4,2,1 4,3,2,1

4. 有种

5. 略

6. 算法：  
#define MaxSize 100

int PAIRBRACKET(char E[])

{

char STACK[MaxSize];

int i=0,top=-1;

while(E[i]!='@'){

if(E[i]=='('){

STACK[++top]=E[i];

}

if(E[i]==')'){

if(top==-1)

return 0;

top--;

}

i++;

}

return top ==-1;

}

7. 略

8. 算法：void CHANGE(int num,int r)

{

STLink p,top=NULL;

do{

p=(STLink)malloc(sizeof(STNode));

p->data=num%r;

p->link=top;

top=p;

num=num/r;

}while(num!=0);

while(top!=NULL){

printf("%d", top->data);

p=top;

top=top->link;

free(p);

}

}

9. 算法：

#define MaxSize 500

int FINDVAL(int n)

{

int STACK[MaxSize][3],top;

int fval;

top=0;

STACK[top][1]=n;

while(n!=0){

n=n/2;

STACK[++top][1]=n;

}

STACK[top][0]=1;

while(top>0){

fval=STACK[top--][0];

STACK[top][2]=fval;

STACK[top][0]=STACK[top][1]\*STACK[top][2];

}

return STACK[top][0];

}

10. 算法：

(1). 递归算法：

int ACK(int m,int n)

{

if(m==0)

return n+1;

else if(m==0)

return ACK(m-1,1);

else

return ACK(m-1,ACK(m,n-1));

}

(2). 非递归算法

#define MaxSize 70000

int ACK(int m,int n)

{

int STACK[MaxSize],top=0;

STACK[top]=m;

STACK[top+1]=n;

while(top>=0){

if(STACK[top]>0)

if(STACK[top+1]>0){

STACK[top+2]=STACK[top+1]-1;

STACK[top+1]=STACK[top];

STACK[top]=STACK[top++]-1;

}else{

STACK[top]=STACK[top-1];

STACK[top+1]=1;

}

else{

STACK[top]=STACK[top+1]+1;

top--;

}

return STACK[0];

}

}

(3). ACK(2,1)=STACK[0]=5

11. 算法：

#define MaxSize 10000

int GCD(int m,int n)

{

int temp,STACK[MaxSize][2],top=0;

STACK[top][0]=m;

STACK[top][1]=n;

while(STACK[top][1]!=0)

if(STACK[top][0]<STACK[top][1]){

temp = STACK[top++][0];

STACK[top][0]=STACK[top-1][1];

STACK[top][1]=temp;

}else{

temp=STACK[top][0]%STACK[top++][1];

STACK[top][0]=STACK[top-1][1];

STACK[top][1]=temp;

}

return STACK[top][0];

}

12. 略

13. 略

14. 略

15. 算法：

void HANOI(int n,char A,char B,char C)

{

if(n==1)

//MOVE(A,C)表示将盘片A移到C

MOVE(A,C);

else{

HANOI(n-1,A,C,B);

MOVE(A,C);

HANOI(n-1,B,A,C);

}

}