.假设正读和反读都相同的字符序列为“回文”，例如，‘abba’和‘abcba’是回文，‘abcde’ 和‘ababab’则不是回文。假设一字符序列已存入计算机，请分析用线性表、堆栈和队列等方式正确输出其回文的可能性？  
答：线性表是随机存储，可以实现，靠循环变量（j--）从表尾开始打印输出；  
堆栈是后进先出，也可以实现，靠正序入栈、逆序出栈即可；  
队列是先进先出，不易实现。  
哪种方式最好，要具体情况具体分析。若正文在机内已是顺序存储，则直接用线性表从后往前读取即可，或将堆栈栈顶开到数组末尾，然后直接用POP动作实现。（但堆栈是先减后压还是……）  
若正文是单链表形式存储，则等同于队列，需开辅助空间，可以从链首开始入栈，全部压入后再依次输出。  
  
  
4.顺序队的“假溢出”是怎样产生的？如何知道循环队列是空还是满？  
答：一般的一维数组队列的尾指针已经到了数组的上界，不能再有入队操作，但其实数组中还有空位置，这就叫“假溢出”。  
采用循环队列是解决假溢出的途径。  
另外，解决队满队空的办法有三：  
① 设置一个布尔变量以区别队满还是队空；  
② 浪费一个元素的空间，用于区别队满还是队空。  
③ 使用一个计数器记录队列中元素个数（即队列长度）。  
我们常采用法②，即队头指针、队尾指针中有一个指向实元素，而另一个指向空闲元素。  
判断循环队列队空标志是： f=rear      队满标志是：f=(r+1)%N  
  
5.设循环队列的容量为40（序号从0到39），现经过一系列的入队和出队运算后，有  
① front=11，rear=19;    ② front=19，rear=11；问在这两种情况下，循环队列中各有元素多少个？  
答：用队列长度计算公式：  (N＋r－f)% N  
① L=（40＋19－11）% 40=8               ② L=（40＋11－19）% 40=32  
  
  
五、阅读理解（每小题5分，共20分。至少要写出思路）  
1. 按照四则运算加、减、乘、除和幂运算（↑）优先关系的惯例，并仿照教材例3-2的格式，画出对下列算术表达式求值时操作数栈和运算符栈的变化过程：  
A－B×C/D+E↑F  
答：   
  
  
2. 写出下列程序段的输出结果（栈的元素类型SElem Type为char）。  
void main( ){  
Stack S;  
Char x,y;  
InitStack(S);  
X=’c’;y=’k’;  
Push(S,x); Push(S,’a’);  Push(S,y);  
Pop(S,x); Push(S,’t’); Push(S,x);  
Pop(S,x); Push(S,’s’);  
while(!StackEmpty(S)){ Pop(S,y);printf(y); };  
Printf(x);  
}  
答：输出为“stack”。  
  
3. 写出下列程序段的输出结果（队列中的元素类型QElem Type为char）。  
void main( ){  
Queue Q;  Init Queue (Q);  
Char x=’e’; y=’c’;  
EnQueue (Q,’h’); EnQueue (Q,’r’);  EnQueue (Q, y);  
DeQueue (Q,x); EnQueue (Q,x);   
DeQueue (Q,x); EnQueue (Q,’a’);   
while(!QueueEmpty(Q)){ DeQueue (Q,y);printf(y); };  
Printf(x);  
}  
答：输出为“char”。  
  
4. 简述以下算法的功能（栈和队列的元素类型均为int）。  
void algo3(Queue &Q){  
Stack S; int d;  
InitStack(S);  
while(!QueueEmpty(Q)){   
DeQueue (Q,d);  Push(S,d);  
};  
while(!StackEmpty(S)){   
Pop(S,d); EnQueue (Q,d);   
}  
}  
答：该算法的功能是：利用堆栈做辅助，将队列中的数据元素进行逆置。  
  
  
  
六、算法设计（每小题5分，共15分。至少要写出思路）  
1. 假设一个算术表达式中包含圆括弧、方括弧和花括弧三种类型的括弧，编写一个判别表达式中括弧是否正确配对的函数correct(exp,tag)；其中：exp为字符串类型的变量（可理解为每个字符占用一个数组元素），表示被判别的表达式，tag为布尔型变量。  
答：用堆栈st进行判定，将 ( 、 [ 或 { 入栈，当遇到 }  、 ]  或 ) 时，检查当前栈顶元素是否是对应的( 、 [ 或 {，若是则退栈，否则返回表示不配对。当整个算术表达式检查完毕时，若栈为空表示括号正确配对，否则不配对。  
编程后的整个函数如下（李书P31—32）  
#define m0 100     /\*m0为算术表达式中最多字符个数\*/  
correct(exp,tag)  
char exp[m0];  
int tag;  
{char st[m0];  
int top=0, i=1;  
tag=1;  
while (i<=m0 && tag)  
{if (exp= = ‘(‘||exp= =’[‘||exp= =’{‘）   /\*遇到‘(‘、’[‘或’{‘，则将其入栈\*/  
{top++;  
st=exp;  
}  
if (exp= =’)’ )   /\*遇到’)’ ，若栈顶是‘(‘，则继续处理，否则以不配对返回\*/  
  if(st= =‘(‘ ) top--;  
else tag=0;  
if (exp= =’ )’ )   /\*遇到’ ]’ ，若栈顶是‘[‘，则继续处理，否则以不配对返回\*/  
  if(st= =‘[ ‘]  top--;  
else tag=0;  
if (exp= =’)’ )   /\*遇到’ }’ ，若栈顶是‘{‘，则继续处理，否则以不配对返回\*/  
  if(st= =‘{‘ top--;  
else tag=0;  
i++;  
}  
if(top>0)tag=0;  /\*若栈不空，则不配对\*/  
}  
  
严题集对应答案：  
3.19   
Status AllBrackets\_Test(char \*str)//判别表达式中三种括号是否匹配   
{   
  InitStack(s);   
  for(p=str;\*p;p++)   
  {   
    if(\*p=='('||\*p=='['||\*p=='{') push(s,\*p);   
    else if(\*p==')'||\*p==']'||\*p=='}')   
    {   
      if(StackEmpty(s)) return ERROR;   
      pop(s,c);   
      if(\*p==')'&&c!='(') return ERROR;   
      if(\*p==']'&&c!='[') return ERROR;   
      if(\*p=='}'&&c!='{') return ERROR; //必须与当前栈顶括号匹配   
    }   
  }//for   
  if(!StackEmpty(s)) return ERROR;   
  return OK;   
}//AllBrackets\_Test   
  
答案（已上机通过）  
  
#include<stdio.h>  
#include<stdlib.h>  
void push(char x);  
void pop();  
void correct(enum Boolean &tag);  
//原来的定义是void correct(struct Stack\* head,enum Boolean &tag);  
  
typedef struct Stack  
{  
char data;  
struct Stack \*next;  
};  
  
struct Stack \*head,\*p;  
enum Boolean{FALSE,TRUE}tag;  
  
  
void main()  
{  
head=(struct Stack\*)malloc(sizeof(struct Stack));  
head->data='S';  
head->next=NULL;  
// head's data has not been initialized!!   
correct(tag);  
if(tag)  
printf("Right!");  
else  
printf("Wrong!");  
}  
  
void push(char x)  
{  
p=(struct Stack\*)malloc(sizeof(struct Stack));  
if(!p)  
printf("There's no space.\n");  
else  
{  
p->data=x;  
p->next=head;  
head=p;  
}  
  
}  
// if you define the "Correct" function like that  
//Debug will show that the Push action doesn’t take effection  
  
void pop()  
{  
if(head->next==NULL)  
printf("The stack is empty.\n");  
else  
{  
p=head;  
head=head->next;  
free(p);  
}  
}  
  
//void correct(struct Stack\* head,enum Boolean &tag)  
void correct(enum Boolean &tag)  
{  
int i;  
char y;  
  
printf("Please enter a bds:");  
for(i=0;y!='\n';i++)  
{  
scanf("%c",&y);  
if((y==')'&&head->data=='(')||(y==']'&&head->data=='[')||(y=='}'&&head->data=='{'))  
pop();  
else if((y=='(')||(y=='[')||(y=='{'))  
push(y);  
/\*调试程序显示，y并没有被推入堆栈中。即head->data的值在Push中显示为y的值，但是出Push函数。马上变成Null。\*/  
else  
continue;  
}  
  
if(head->next==NULL)       //原来的程序是if(head ==NULL) tag=TRUE;  
tag=TRUE;  
else  
tag=FALSE;  
}  
  
/\*总结： 由于head为全局变量，所以不应该将其再次作为函数的变量。因为C语言的函数变量是传值机制，所以在函数中对参数进行了拷贝复本，所以不能改变head的数值。\*/