1.8

1. n-1
2. n-1
3. n-1
4. n\*(n+1)/2
5. 
6. 若n<=0,运行0次，否则n次
7. √n向下取整
8. 1100

1.11

1. 1<=n<=40
2. 1<=n<=16
3. O(2^n)更适合
4. n=61时会反超

2.7

1)11.3.14

2)10.12.8.3???

3)10.12.7.3.14

4)12.11.3.14

1. 如果尾元结点指的是倒数第二个节点：12.9.11.3.14

如果尾元结点指的是最后一个节点：12.9.1.11.3.14

2.12

void Compare(Sqlist A,Sqlist B)

{

int i,j,n;

if(A.length<B.length)

n=A.length;

else

n=B.length;

for(i=0;i<n;i++)

{

if(A.elem[i]>B.elem[i])

{

j=1;

break;

}

if(A.elem[i]<B.elem[i])

{

j=-1;

break;

}

}

if(j==0)

{

if(A.length>n)

j=1;

if(B.length>n)

j=-1;

if(A.length==B.length)

j=0;

}

if(j==1)

cout<<”A>B”<<endl;

if(j==0)

cout<<”A=B”<<endl;

if(j==-1)

cout<<”A<B”<<endl;

}

3.1

1）321.123.213.231.132

2）435612不可以，因为若是4356已经出栈，那么12必然已经压栈，所以2必须在1之 前出栈。

135426可以，S:1 X:1 S:2,3 X:3 S:4,5 X:5,4,2 S:6 X:6

3.6

1. 若是i单独进栈单独出栈，然后j,k进栈出栈，会有ijk,ikj两种情况
2. 若是ij进栈，一个或者两个一起出栈，然后k进栈，两个或者一个出栈，会有jki,jik两种情况
3. 若是ijk同时进栈出栈，会有kji一种情况
4. 总共只有以上五种情况，故不会出现jik这种情况。

3.17

bool Distinguish(char a[])

{

int i=0;

Sqstack s;

SqQueue q;

Elemtype a,b;

InitSqStack(s);

InitSqQueue(q);

while(a[i]!='&')

{

enQueue(q,a[i++]);

}//前半部分入队列

i++;

while(a[i]==’@’)

{

enstack(s,a[i++]);

}//后半部分入栈

if(s.length!=q.length)

return false;

//若长度不等无须比较

for(i=0;i<s.length;i++)

{

a=pop(s);

b=deQueue(q);

if(a!=b)

return false;

}//逐个比较

return true;

}

3.29

typedef struct SqQueue

{

Elemtype \*base;

int first;

int rear;

bool tag;

};//队列结构体

void InitSqQueue(SqQueue &q)

{

q.base=new DataType[5];

q.first=0;

q.rear=0;

q.tag=0;

}//建立空队列

bool enSqQueue(SqQueue &q, DataType e)

{

if(q.first=q.rear&&q.tag)

return false;

else

{

\*(q.base+q.rear)=e;

q.rear=(q.rear+1)%5;

if(q.first==q.rear)

q.tag=1;

return true;

}

}//入队列

bool DeSqQueue(SqQueue &q,DataType &e)

{

if(q.first=q.rear&&q.tag)

return false;

else

{

e=\*(q.base+q.first);

q.first=(q.first+1)%5;

if(q.first==q.rear)

q.tag=0;

return true;

}

}//出队列