1. 以下程序段的输出结果为 64 ，用大”O”记号表示其时间复杂度为: 。

**int** n=16;

**int** k = 0;

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

**for** (**int** j=1;j<n;j=j<<1)

k++;

cout<<k;

2．Suppose we have implemented the Stack ADT as a singly-linked-list with head and tail pointers and no sentinels. Which of the following best describe the running times for the functions push and pop, assuming there are n items in the list, and that the bottom of the stack is at the head of the list (all pushing and popping occurs at the tail)?

(a) O(1) for both functions.

(b) O(n) for both functions.

(c) O(1) for push and O(n) for pop.

(d) O(n) for push and O(1) for pop.

(e) None of these is the correct choice.

3. 设计“猜猜我的QQ号”算法。（直接使用第2-4章中所学数据结构及方法。）

具体描述：给定一个数字串source和以下规则，得出一个qq号，以字符串形式返回。

规则：

对source串进行处理，

首先将第1个数删除，接着将第2个数放到这串数的末尾，

再将第3个数删除并将第4个数放到这串数的末尾，

再将第5个数删除……

直到剩下最后1个数，将最后1个数也删除。按照刚才删除的顺序，把这些删除的数连在一起就是我的QQ号。

如：给出的数字串是631758924，则返回的qq号是615947283。

string getQQ(string numbers){

Extended\_queue source,dest;

char ch;

for (int i=0;i<numbers.size();i++)

source.append(numbers[i]);

while (!source.empty()){

source.serve\_and\_retrieve(ch);

dest.append(ch);

if (!source.empty()){

source.serve\_and\_retrieve(ch);

source.append(ch);

}

}

string result;

while(!dest.empty()){

dest.serve\_and\_retrieve(ch);

result+=ch;

}

return result;

}