-1　银行业务队列简单模拟　(25　分)

设某银行有A、B两个业务窗口，且处理业务的速度不一样，其中A窗口处理速度是B窗口的2倍 —— 即当A窗口每处理完2个顾客时，B窗口处理完1个顾客。给定到达银行的顾客序列，请按业务完成的顺序输出顾客序列。假定不考虑顾客先后到达的时间间隔，并且当不同窗口同时处理完2个顾客时，A窗口顾客优先输出。

输入格式:

输入为一行正整数，其中第1个数字N(≤1000)为顾客总数，后面跟着N位顾客的编号。编号为奇数的顾客需要到A窗口办理业务，为偶数的顾客则去B窗口。数字间以空格分隔。

输出格式:

按业务处理完成的顺序输出顾客的编号。数字间以空格分隔，但最后一个编号后不能有多余的空格。

输入样例:

```
8 2 1 3 9 4 11 13 15
```

输出样例:

```
1 3 2 9 11 4 13 15
```

```
#include<stdio.h>
#include<stdlib.h>
#define OK 0
#define SIZE 100
#define ADD_SIZE 100
#define error -1

typedef int Element;
typedef int status;

typedef struct {
  Element* base;
  int front;
  int rear;
}Queue;

status init(Queue &B,int temp){
    B.base=(Element*)malloc(temp*sizeof(Element*));
    B.front=B.rear=0;
    return OK;
}

status add(Queue &B,Element E){
    if((B.rear+1)%SIZE==B.front) return error;
    B.base[B.rear]=E;
    B.rear=(B.rear+1)%SIZE;
    return OK;
}
```

```
status remove(Queue &B,Element E,int num){
    if(B.front==B.rear) return error;
    E=B.base[B.front];
    B.front=(B.front+1)%SIZE;
    if(num!=1){
    printf("%d ",E);
    }else{
        printf("%d",E);
    }
    return E;
}

int main(){
    int N;
   if(scanf("%d",&N)==1){};
   if(N>=1000){
     return 0;
   }
   Queue QQ,Q;
   init(Q,N);
   init(QQ,N);
   int temp;

   for(int i=0;i<N;i++){
      if(scanf("%d",&temp)==1){};
      if(temp%2!=0){
          add(QQ,temp);
          //QQ.base[QQ.rear++]=temp;
      }else{
          add(Q,temp);
          //Q.base[Q.rear++]=temp;
      }

   }
  int E=0;
   while(QQ.front!=QQ.rear&&Q.front!=Q.rear){
       for(int k=0;k<2;k++){
          if(QQ.front!=QQ.rear){
                remove(QQ,E,N);
                N--;
          }else{
              break;
          }
       }
       for(int j=0;j<1;j++){
          if(Q.front!=Q.rear){
```

```
            remove(Q,E,N);
            N--;
        }else{
            break;
        }
    }

}

while(Q.front!=Q.rear){


        if(Q.front!=Q.rear){
            remove(Q,E,N);
            N--;
        }else{
            break;
        }

}


while(QQ.front!=QQ.rear){


        if(QQ.front!=QQ.rear){
            remove(QQ,E,N);
            N--;
        }else{
            break;
        }

}

return 0;

}
```