整理了网上２０１４华为机试题，近几十个网页的资料，同时本人对每道题均进行了验证。并且对一些题目进行了更好的修正。希望对后来者有所帮助。

1．第一题的题目大概是输入整型数组求数组的最小数和最大数之和，例如输入1,2,3,4则输出为5，当输入只有一个数的时候，则最小数和最大数都是该数，例如只输入1，则输出为2；另外数组的长度不超过50

#include<stdio.h>

main()

{

int num[50]={0};

int i,n;

printf("请输入整型数组的长度(1~50)：");

scanf("%d",&n);

printf("请输入整型数组的元素：");

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

{

scanf("%d",&num[i]);

}

int min\_num=num[0];

int max\_num=num[0];

for(int j=0;j<n;j++)

{

if(max\_num<num[j])

max\_num=num[j];

else if(min\_num>num[j])

min\_num=num[j];

}

int sum=min\_num+max\_num;

printf("数组中最大与最小值之和：%d\n",sum);

return 0;

}

2．求两个长长整型的数据的和并输出，例如输入1233333333333333 。。。 3111111111111111111111111.。。。，则输出。。。。

#include<stdio.h>

#include<string.h>

#include<malloc.h>

main()

{

char \*num1,\*num2; //两个长长整型数据

char \*sum;

// int temp;

int len\_num1,len\_num2; // 两个长长整型数据的长度

int len\_max,len\_min;

num1=(char\*)malloc(sizeof(char));

num2=(char\*)malloc(sizeof(char));

printf("输入两个长长整型数据：");

scanf("%s",num1);

printf("输入两个长长整型数据：");

scanf("%s",num2);

len\_num1=strlen(num1);

len\_num2=strlen(num2);

len\_max=(len\_num1>=len\_num2)? len\_num1:len\_num2;

len\_min=(len\_num1<=len\_num2)? len\_num1:len\_num2;

int len\_max1=len\_max;

sum=(char\*)malloc(sizeof(char)\*len\_max);

memset(sum,0x00,len\_max+1);//切忌初始化

for(;len\_num1>0&&len\_num2>0;len\_num1--,len\_num2--)

{

sum[len\_max--]=((num1[len\_num1-1]-'0')+(num2[len\_num2-1]-'0'));

}

if(len\_num1>0)

{

sum[len\_max--]=num1[len\_num1 - 1 ]-'0';

len\_num1--;

}

if(len\_num2>0)

{

sum[len\_max--]=num1[len\_num2 - 1]-'0';

len\_num2--;

}

for(int j=len\_max1;j>=0;j--) //实现进位操作

{

// temp=sum[j]-'0';

if(sum[j]>=10)

{

sum[j-1]+=sum[j]/10;

sum[j]%=10;

}

}

char \*outsum=(char\*)malloc(sizeof(char)\*len\_max1);

j=0;

while(sum[j]==0) //跳出头部0元素

j++;

for(int m=0;m<len\_max1;j++,m++)

outsum[m]=sum[j]+'0';

outsum[m]='\0';

printf("输出两长长整型数据之和:%s\n",outsum);

return 0;

}

3.通过键盘输入一串小写字母(a~z)组成的字符串。请编写一个字符串过滤程序，若字符串中出现多个相同的字符，将非首次出现的字符过滤掉。

比如字符串“abacacde”过滤结果为“abcde”。

要求实现函数：

void stringFilter(const char \*pInputStr, long lInputLen, char \*pOutputStr);

【输入】 pInputStr： 输入字符串

lInputLen： 输入字符串长度

【输出】 pOutputStr： 输出字符串，空间已经开辟好，与输入字符串等长；

#include <stdio.h>

#include<string.h>

#include<malloc.h>

void stringFilter(const char \*p\_str, long len, char \*p\_outstr)

{

int array[256]={0};

const char \*tmp = p\_str;

for(int j=0;j<len;j++)

{

if(array[tmp[j]]==0)

\*p\_outstr++= tmp[j];

array[tmp[j]]++;

}

\*p\_outstr = '\0';

}

void main()

{

char \*str = "cccddecc";

int len = strlen(str);

char \* outstr = (char \*)malloc(len\*sizeof(char));

stringFilter(str,len,outstr);

printf("%s\n",outstr);

free(outstr);

outstr = NULL;

}

4.通过键盘输入一串小写字母(a~z)组成的字符串。请编写一个字符串压缩程序，将字符串中连续出席的重复字母进行压缩，并输出压缩后的字符串。

压缩规则：

1. 仅压缩连续重复出现的字符。比如字符串"abcbc"由于无连续重复字符，压缩后的字符串还是"abcbc".

2. 压缩字段的格式为"字符重复的次数+字符"。例如：字符串"xxxyyyyyyz"压缩后就成为"3x6yz"

要求实现函数：

void stringZip(const char \*pInputStr, long lInputLen, char \*pOutputStr);

【输入】 pInputStr： 输入字符串

lInputLen： 输入字符串长度

【输出】 pOutputStr： 输出字符串，空间已经开辟好，与输入字符串等长；

#include <stdio.h>

#include<string.h>

#include<malloc.h>

void stringZip(const char \*p\_str, long len, char \*p\_outstr)

{

int count=1;

for(int i=0;i<len;i++)

{

if(p\_str[i]==p\_str[i+1])

{

count++;

}

else

{

if(count>1)

{

\*p\_outstr++ = count +'0';

\*p\_outstr++ =p\_str[i];

}

else

{

\*p\_outstr++ =p\_str[i];

}

count = 1;//注意其位置

}

}

\*p\_outstr = '\0';

}

void main()

{

char \*str = "cccddecc";

printf("压缩之前的字符串为：%s\n",str);

int len = strlen(str);

char \* outstr = (char\*)malloc(len\*sizeof(char));

stringZip(str,len,outstr);

printf("压缩之后的字符串为：%s\n",outstr);

free(outstr);

outstr = NULL;

}

5.通过键盘输入100以内正整数的加、减运算式，请编写一个程序输出运算结果字符串。

输入字符串的格式为：“操作数1 运算符 操作数2”，“操作数”与“运算符”之间以一个空格隔开。

补充说明：

1. 操作数为正整数，不需要考虑计算结果溢出的情况。

2. 若输入算式格式错误，输出结果为“0”。

要求实现函数：

void arithmetic(const char \*pInputStr, long lInputLen, char \*pOutputStr);

【输入】 pInputStr： 输入字符串

lInputLen： 输入字符串长度

【输出】 pOutputStr： 输出字符串，空间已经开辟好，与输入字符串等长；

#include <stdio.h>

#include<string.h>

#include<stdlib.h>

void arithmetic(const char \*input, long len, char \*output)

{

char s1[10];

char s2[10];

char s3[10];

int cnt = 0;

int len\_input=strlen(input);

for(int i=0;i<len\_input;++i)

{

if(input[i]==' ')

cnt++;

}

if(cnt!=2)

{

\*output++ = '0';

\*output = '\0';

return;

}

sscanf(input,"%s %s %s",s1,s2,s3);

if(strlen(s2)!=1||(s2[0]!='+'&&s2[0]!='-'))

{

\*output++ = '0';

\*output = '\0';

return;

}

int len\_s1=strlen(s1);

for(i=0;i<len\_s1;i++)

{

if(s1[i]<'0'||s1[i]>'9')

{

\*output++ = '0';

\*output = '\0';

return;

}

}

int len\_s3=strlen(s3);

for(i=0;i<len\_s3;i++)

{

if(s3[i]<'0'||s3[i]>'9')

{

\*output++ = '0';

\*output = '\0';

return;

}

}

int x = atoi(s1);

int y = atoi(s3);

if(s2[0]=='+')

{

int result = x+y;

itoa(result,output,10);

}

else if(s2[0]=='-')

{

int result = x-y;

itoa(result,output,10);

}

else

{

\*output++ = '0';

\*output = '\0';

return;

}

}

void main()

{

char str[] = {"10 - 23"};

char outstr[10];

int len = strlen(str);

arithmetic(str,len,outstr);

printf("%s\n",str);

printf("%s\n",outstr);

}

6.一组人（n个），围成一圈，从某人开始数到第三个的人出列，再接着从下一个人开始数，最终输出最终出列的人

（约瑟夫环是一个数学的应用问题：已知n个人（以编号1，2，3...n分别表示）围坐在一张圆桌周围。从编号为k的人开始报数，数到m的那个人出列；他的下一个人又从1开始报数，数到m的那个人又出列；依此规律重复下去，直到圆桌周围的人全部出列。）

#include <stdio.h>

#include<string.h>

#include<stdlib.h>

#include<malloc.h>

typedef struct Node

{

int data;

struct Node \*next;

}LinkList;

LinkList \*create(int n)

{

LinkList \*p,\*q,\*head;

int i=1;

p=(LinkList\*)malloc(sizeof(LinkList));

p->data=i;

head=p;

for(i=1;i<=n;i++)

{

q=(LinkList\*)malloc(sizeof(LinkList));

q->data=i+1;

p->next=q;

p=q;

}

p->next=head; //使链表尾连接链表头，形成循环链表

return head;

free(p);

p=NULL;

free(q);

q=NULL;

}

void deletefun(LinkList \*L,int m)

{

LinkList \*p,\*q,\*temp;

int i;

p=L;

while(p->next!=p)

{

for(i=1;i<m;i++)

{

q=p;

p=p->next;

}

printf("%5d",p->data);

temp=p;

q->next=p->next;

p=p->next;

free(temp);

}

printf("%5d\n",p->data);

}

int main()

{

int n=7,m=3;

LinkList \*head1;

head1=create(n);

deletefun(head1,m);

return 0;

}

7..输入一串字符，只包含“0-10”和“，”找出其中最小的数字和最大的数字（可能不止一个），输出最后剩余数字个数。如 输入 “3,3,4,5,6,7,7”

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void main()

{

char str[100];

printf("输入一组字符串：\n");

scanf("%s",&str);

int len=strlen(str);

int array[100];

int count=0;

for(int i=0;i<len;i++)

{

if(str[i]>='0'&&str[i]<='9')

array[count++]=str[i]-'0';

}

array[count]='\0';

int result=count;

int min=array[0];

int max=array[0];

for(int j=0;j<count;j++)

{

if(max<array[j])

max=array[j];

else if(min>array[j])

min=array[j];

}

for(int k=0;k<count;k++)

{

if(array[k]==min)

result--;

if(array[k]==max)

result--;

}

printf("%d\n",result);

}

8.输入一组身高在170到190之间（5个身高），比较身高差，选出身高差最小的两个身高；若身高差相同，选平均身高高的那两个身高；从小到大输出；  
如 输入 170 181 173 186 190 输出 170 173

#include<stdio.h>

#include<stdlib.h>

#define N 5

int main()

{

int Height[N];

int dmin;

int H1,H2;

int i,j,temp;

printf("请输入一组身高在170到190之间的数据（共5个）:\n");

for(int k=0;k<N;k++)

scanf("%d",&Height[k]);

printf("\n");

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

for(j=1;j<N-i&&Height[j-1]>Height[j];j++)

{

temp=Height[j-1];

Height[j-1]=Height[j];

Height[j]=temp;

}

H1=Height[0];

H2=Height[1];

dmin=H2-H1;

for(int m=2;m<N;m++)

{

if(Height[m]-Height[m-1]<=dmin)

{

H1=Height[m-1];

H2=Height[m];

dmin=Height[m]-Height[m-1];

}

}

printf("身高差最小的两个身高为:\n");

printf("%d,%d\n",H1,H2);

return 0;

}

9. **删除子串**，只要是原串中有相同的子串就删掉，不管有多少个，返回子串个数。

#include <stdio.h>

#include <stdlib.h>

#include <assert.h>

#include <string.h>

int delete\_sub\_str(const char \*str,const char \*sub\_str,char \*result)

{

assert(str != NULL && sub\_str != NULL);

const char \*p,\*q;

char \*t,\*temp;

p = str;

q = sub\_str;

t = result;

int n,count = 0;

n = strlen(q);

temp = (char \*)malloc(n+1);

memset(temp,0x00,n+1);

while(\*p)

{

memcpy(temp,p,n);

if(strcmp(temp,q) == 0 )

{

count++;

memset(temp,0x00,n+1);

p = p + n;

}

else

{

\*t = \*p;

p++;

t++;

memset(temp,0x00,n+1);

}

}

free(temp);

return count;

}

void main()

{

char s[100] = {‘\0’};

int num = delete\_sub\_str(“123abc12de234fg1hi34j123k”,”123”,s);

printf(“The number of sub\_str is %d\r\n”,num);

printf(“The result string is %s\r\n”,s);

}

10. 要求编程实现上述高精度的十进制加法。要求实现函数：

void add (const char \*num1, const char \*num2, char \*result)

【输入】num1：字符串形式操作数1，如果操作数为负，则num1[0]为符号位'-'

num2：字符串形式操作数2，如果操作数为负，则num2[0]为符号位'-'

【输出】result：保存加法计算结果字符串，如果结果为负，则result[0]为符号位。

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void move(char \*str, int length) //移除字母前的"-"符号

{

if(str[0] != '-')

return;

int i;

for(i = 0; i < length-1; i++)

str[i] = str[i+1];

str[i] = '\0';

}

int remove\_zero(char \*result, int length)

{

int count = 0;

for(int i = length-1; i > 0; i--) //从最后开始移除0，直到遇到非0数字，只对最初位置上的0不予判断

{

if(result[i] == '0')

{

result[i] = '\0';

count++;

}else

return length-count;

}

return length - count;

}

void reverse(char \*result, int length) //将字符串倒转

{

char temp;

for(int i = 0; i <= (length-1)/2; i++)

{

temp = result[i];

result[i] = result[length-1-i];

result[length-1-i] = temp;

}

}

int real\_add(char \*str1, char \*str2, char \*result, const bool flag)

{

int len1 = strlen(str1);

int len2 = strlen(str2);

int n1, n2, another = 0; //another表示进位

int cur\_rs = 0; //表示result的当前位数

int i, j;

int curSum;

for(i = len1-1, j = len2-1; i >= 0 && j >= 0; i--, j--)

{

n1 = str1[i] - '0';

n2 = str2[j] - '0';

curSum = n1 + n2 + another;

result[cur\_rs++] = curSum % 10 + '0';

another = curSum / 10;

}

if(j < 0)

{

while(i >= 0) //遍历str1剩余各位

{

n1 = str1[i--] - '0';

curSum = n1 + another;

result[cur\_rs++] = curSum % 10 + '0';

another = curSum / 10;

}

if(another != 0) //如果还有进位未加上

result[cur\_rs++] = another + '0';

}

else

{

while(j >= 0)

{

n2 = str2[j--] - '0';

curSum = n2 + another;

result[cur\_rs++] = curSum % 10 + '0';

another = curSum / 10;

}

if(another != 0)

result[cur\_rs++] = another + '0';

}

result[cur\_rs] = '\0';

cur\_rs = remove\_zero(result, cur\_rs);

if(!flag)

{

result[cur\_rs++] = '-';

result[cur\_rs] = '\0';

}

reverse(result, strlen(result));

return cur\_rs;

}

int real\_minus(char \*str1, char \*str2, char \*result) //使用str1减去str2

{

char big[100], small[100];

int big\_len, sml\_len;

int len1 = strlen(str1);

int len2 = strlen(str2);

bool flag = false; //用于标记str2是否比str1大

if(len1 < len2)

flag = true;

else if(len1 == len2)

{

if(strcmp(str1, str2) == 0)

{

result[0] = '0';

result[1] = '\0';

return 1;

}else if(strcmp(str1,str2) < 0)

flag = true;

}

if(flag) //将str1和str2交换，确保str1指向的值是其中较大者，最后通过flag确定要不要给前面加-号

{

char \*temp = str1;

str1 = str2;

str2 = temp;

len1 = strlen(str1);

len2 = strlen(str2);

}

int n1, n2, another = 0; //another表示是否有借位

int i, j;

int cur\_rs = 0;

int curMinus;

for(i = len1-1, j = len2-1; i>=0 && j>=0; i--,j--)

{

n1 = str1[i] - '0';

n2 = str2[j] - '0';

if(n1 >= n2+another)

{

result[cur\_rs++] = (n1-n2-another) +'0';

another = 0;

}

else

{

result[cur\_rs++] = (n1+10-n2-another) + '0';

another = 1;

}

}

while(i >= 0)

{

n1 = str1[i--] - '0';

if(another != 0)

{

n1 -= another;

another = 0;

}

result[cur\_rs++] = n1 + '0';

}

result[cur\_rs] = '\0';

cur\_rs = remove\_zero(result, cur\_rs);

if(flag)

{

result[cur\_rs++] = '-';

result[cur\_rs] = '\0';

}

reverse(result, cur\_rs);

return cur\_rs;

}

void addi(const char \*num1, const char \*num2, char \*result)

{

int len1 = strlen(num1);

int len2 = strlen(num2);

int rs\_len;

if(!len1 || !len2)

return;

char str1[100], str2[100];

strncpy(str1, num1, len1);

str1[len1] = '\0';

strncpy(str2, num2, len2);

str2[len2] = '\0';

if(str1[0] == '-' && str2[0] == '-')

{

move(str1, len1);

move(str2, len2);

rs\_len = real\_add(str1, str2, result, false);

}else if(str1[0] == '-')

{

move(str1, len1);

rs\_len = real\_minus(str2, str1, result);

}

else if(str2[0] == '-')

{

move(str2, len2);

rs\_len = real\_minus(str1, str2, result);

}else

rs\_len = real\_add(str1, str2, result, true);

}

//int main(int argc, char \*argv[])

int main()

{

char num1[100],num2[100];

printf("请输入两个整型数据：\n");

scanf("%s%s",num1,num2);

char result[100];

memset(result, 0, 100);

addi(num1,num2, result);

printf("%s\n", result);

return 0;

}

11. **描述：10个学生考完期末考试评卷完成后，A老师需要划出及格线，要求如下：  
(1) 及格线是10的倍数；  
(2) 保证至少有60%的学生及格；  
(3) 如果所有的学生都高于60分，则及格线为60分**

**输入：输入10个整数，取值0~100**

**输出：输出及格线，10的倍数**

#include<stdio.h>

void bubblesort(int arr[])

{

int i,j,temp;

for(i=0;i<10;i++)

for(j=0;j<9-i&&arr[j]>arr[j+1];j++)

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

int GetPassLine(int a[])

{

bubblesort(a);

if(a[0]>=60)

return 60;

else

return (((int)a[4]/10)\*10);

}

main()

{

int a[10]={0};

int result;

printf("请随机输入10个成绩（0-100）：\n");

scanf("%d%d%d%d%d%d%d%d%d%d",&a[0],&a[1],&a[2],&a[3],&a[4],&a[5],&a[6],&a[7],&a[8],&a[9]);

printf("\n");

result=GetPassLine(a);

printf("及格线为:%d\n",result);

return 1;

}

12. **描述：一条长廊里依次装有n(1 ≤ n ≤ 65535)盏电灯，从头到尾编号1、2、3、…n-1、n。每盏电灯由一个拉线开关控制。开始，电灯全部关着。**

**有n个学生从长廊穿过。第一个学生把号码凡是1的倍数的电灯的开关拉一下；接着第二个学生把号码凡是2的倍数的电灯的开关拉一下；接着第三个学生把号码凡是3的倍数的电灯的开关拉一下；如此继续下去，最后第n个学生把号码凡是n的倍数的电灯的开关拉一下。n个学生按此规定走完后，长廊里电灯有几盏亮着。注：电灯数和学生数一致。**

**输入：电灯的数量**

**输出：亮着的电灯数量**

**样例输入：3**

**样例输出：1**

#include<stdio.h>

#define Max\_Bubl\_Num 65535

int GetLightLampNum(int n)

{

int BublNum[Max\_Bubl\_Num]={0}; //0表示灯灭，1表示灯亮

unsigned int i,j;

unsigned int count=0;

for(i=1;i<=n;i++)

for(j=i;j<=n&&j%i==0;j++)

{

BublNum[j-1]+=1;

BublNum[j-1]=BublNum[j-1]%2;

}

for(int k=0;k<n;k++)

{

if(BublNum[k]==1)

count++;

}

return count;

}

int main()

{

int n,result;

printf("请输入灯的数量（1-65535）:\n");

scanf("%d",&n);

result=GetLightLampNum(n);

printf("最后亮灯的数量为:%d\n",result);

return 0;

}

13. **描述：已知2条地铁线路，其中A为环线，B为东西向线路，线路都是双向的。经过的站点名分别如下，两条线交叉的换乘点用T1、T2表示。编写程序，任意输入两个站点名称，输出乘坐地铁最少需要经过的车站数量（含输入的起点和终点，换乘站点只计算一次）。  
地铁线A（环线）经过车站：A1 A2 A3 A4 A5 A6 A7 A8 A9 T1 A10 A11 A12 A13 T2 A14 A15 A16 A17 A18  
地铁线B（直线）经过车站：B1 B2 B3 B4 B5 T1 B6 B7 B8 B9 B10 T2 B11 B12 B13 B14 B15**

**输入：输入两个不同的站名**

**输出：输出最少经过的站数,含输入的起点和终点，换乘站点只计算一次**

**输入样例：A1 A3**

**输出样例：3**

#include<stdio.h>

#include<string>

#include<queue>

#include<vector>

using namespace std;

#define MAX 35

#define SUBWAY\_A 20

#define SUBWAY\_B 15

typedef struct node{

int adjvex;

struct node \*next;

}edgenode;

typedef struct{

char name[10];

bool flag;

edgenode \*link;

}vexnode;

const char subway\_name1[SUBWAY\_A][10]={"A1","A2","A3","A4","A5","A6","A7","A8","A9","T1","A10","A11","A12","A13","T2","A14","A15","A16","A17","A18"};

const char subway\_name2[SUBWAY\_B][10]={"B1","B2","B3","B4","B5","B6","B7","B8","B9","B10","B11","B12","B13","B14","B15"};

void creat(vexnode ga[]){

int i;

edgenode \*p;

for(i=0;i<MAX;i++){

ga[i].link=NULL;

ga[i].flag=true;

if(i<SUBWAY\_A) strcpy(ga[i].name,subway\_name1[i]);

else strcpy(ga[i].name,subway\_name2[i-20]);

}

//A地铁建邻接表

for(i=1;i<SUBWAY\_A-1;i++){

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=i-1;

p->next=NULL;

ga[i].link=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=i+1;

p->next=NULL;

ga[i].link->next=p;

if(i==9){

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A+4;

p->next=NULL;

ga[i].link->next->next=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A+5;

p->next=NULL;

ga[i].link->next->next->next=p;

}

else if(i==14){

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A+9;

p->next=NULL;

ga[i].link->next->next=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A+10;

p->next=NULL;

ga[i].link->next->next->next=p;

}

}

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A-1;

p->next=NULL;

ga[0].link=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=1;

p->next=NULL;

ga[0].link->next=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A-2;

p->next=NULL;

ga[SUBWAY\_A-1].link=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=0;

p->next=NULL;

ga[SUBWAY\_A-1].link->next=p;

//B地铁建邻接表

for(i=1;i<SUBWAY\_B-1;i++){

if(i==4||i==5||i==9||i==10) continue;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A+i-1;

p->next=NULL;

ga[i+SUBWAY\_A].link=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A+i+1;

p->next=NULL;

ga[i+SUBWAY\_A].link->next=p;

}

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A+3;

p->next=NULL;

ga[SUBWAY\_A+4].link=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=9;

p->next=NULL;

ga[SUBWAY\_A+4].link->next=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=9;

p->next=NULL;

ga[SUBWAY\_A+5].link=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A+6;

p->next=NULL;

ga[SUBWAY\_A+5].link->next=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A+8;

p->next=NULL;

ga[SUBWAY\_A+9].link=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=14;

p->next=NULL;

ga[SUBWAY\_A+9].link->next=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=14;

p->next=NULL;

ga[SUBWAY\_A+10].link=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A+11;

p->next=NULL;

ga[SUBWAY\_A+10].link->next=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A+1;

p->next=NULL;

ga[SUBWAY\_A].link=p;

p=(edgenode\*)malloc(sizeof(edgenode));

p->adjvex=SUBWAY\_A+SUBWAY\_B-2;

p->next=NULL;

ga[SUBWAY\_A+SUBWAY\_B-1].link=p;

// 打印各邻接节点

for(i=0;i<MAX;i++){

printf("%s:",ga[i].name);

edgenode \*s;

s=ga[i].link;

while(s!=NULL){

printf("->%s",ga[s->adjvex].name);

s=s->next;

}

printf("\n");

}

}

int main(){

vexnode ga[MAX];

creat(ga);

int i;

char str[2][10];

while(scanf("%s%s",str[0],str[1])!=EOF){

int temp=0;

for(i=0;i<MAX;i++){

ga[i].flag=true;

if(!strcmp(str[0],ga[i].name)) temp=i;

}

queue<vexnode>q;

q.push(ga[temp]);

ga[temp].flag=false;

int count=0;

int start=0;

int end=1;

bool find\_flag=false;

while(!q.empty()){

if(find\_flag) break;

count++;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("第%d层搜索：",count);

int temp\_end=end;

while(start<temp\_end){

printf("%s ",q.front().name);

if(!strcmp(q.front().name,str[1])){

find\_flag=true;

break;

}

edgenode \*s;

s=q.front().link;

while(s!=NULL){

if(ga[s->adjvex].flag){

q.push(ga[s->adjvex]);

ga[s->adjvex].flag=false;

end++;

//printf("%s ",ga[s->adjvex].name);

}

s=s->next;

}

q.pop();

start++;

}

printf("\n");

}

printf("%d\n",count);

}

return 0;

}

14. 字串转换  
问题描述：  
将输入的字符串（字符串仅包含小写字母‘a’到‘z’），按照如下规则，循环转换后输出：a->b,b->c,…,y->z,z->a；若输入的字符串连续出现两个字母相同时，后一个字母需要连续转换2次。例如：aa 转换为 bc，zz 转换为 ab；当连续相同字母超过两个时，第三个出现的字母按第一次出现算。  
要求实现函数：  
void convert(char \*input,char\* output)  
【输入】 char \*input , 输入的字符串  
【输出】 char \*output ，输出的字符串  
【返回】 无

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

void convert(char \*input,char\* output)

{

if(input==NULL)

return;

char temp='\0';

int len\_input=strlen(input);

int i;

int flag=0;

for(i=0;i<len\_input;i++)

{

if(input[i]!=temp)

{

output[i]=(input[i]-'a'+1)%26+'a';

temp=input[i];

flag=1;

}

else

{

if(flag==1)

{

output[i]=(input[i]-'a'+2)%26+'a';

temp=input[i];

flag=0;

}

else

{

output[i]=(input[i]-'a'+1)%26+'a';

temp=input[i];

flag=1;

}

}

}

output[i]='\0';

}

void main()

{

char \*input="xyz";

char output[256];

// scanf("%s",input);

convert(input,output);

printf("%s\n",output);

}

15. 在给定字符串中找出单词（ “单词”由大写字母和小写字母字符构成，其他非字母字符视为单词的间隔，如空格、问号、数字等等；另外单个字母不算单词）；找到单词后，按照长度进行降序排序，（排序时如果长度相同，则按出现的顺序进行排列），然后输出到一个新的字符串中；如果某个单词重复出现多次，则只输出一次；如果整个输入的字符串中没有找到单词，请输出空串。输出的单词之间使用一个“空格”隔开，最后一个单词后不加空格。  
要求实现函数：  
void my\_word(charinput[], char output[])  
【输入】 char input[], 输入的字符串  
【输出】 char output[]，输出的字符串  
【返回】 无

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

void my\_word(char input[],char output[])

{

char \*p;

char \*temp;

char \*word[10];

int len\_input=strlen(input);

int i,j;

char except[] = ",";

char \*blank = " ";

i=0;

for (i=0;i<len\_input;i++)

{

if (input[i]<'A' || (input[i]>'Z'&&input[i]<'a') || input[i]>'z')

{

input[i]=',';

}

}

j=0;

/\*保存取出的单词\*/

p= strtok(input,except);

while(NULL!=p)

{

word[j++]=p;

p= strtok(NULL,except);

}

for(i=0;i<5;i++)

printf("%s",word[i]);

/\*对单词按照长度降序排序，冒泡法\*/

for (i=0;i<5;i++)

{

for (j=1;j<5-i;j++)

{

if(strlen(word[j-1])<strlen(word[j]))

{

temp=word[j];

word[j]=word[j-1];

word[j-1]=temp;

}

}

}

/\*删除相同单词\*/

for (i=0;i<5;i++)

{

for(j=i+1;j<5;j++)

{

if(strcmp(word[i],word[j])==0)

word[j]="\0";

}

}

/\*将单词连接起来输出\*/

for (j=0;j<5;j++)

{

if (0==j)

{

strncpy(output,word[j],strlen(word[j])+1);

}

else

{

strcat(output,blank);

strcat(output,word[j]);

}

}

return ;

}

int main()

{

char input[] ="some local buses, some1234123drivers";

printf("筛选之前的字符串:%s\n",input);

char output[30];

my\_word(input,output);

printf("筛选之后的字符串:%s",output);

printf("\n");

return 0;

}

１６. 数组中数字都两两相同，只有一个不同，找出该数字：

int findUnique(int\* a, int len)

{

int i = 1;

int temp = a[0];

for(; i < len; i++)

{

temp = temp ^ a[i];

}

printf("%d ", temp);

}

17. 题目二：数组中数字两两相同，有两个不同，找出这两个：

#include <stdlib.h>

int a[] = {1,1,2,4,3,3,2,5};

int findXorSum(int\* a, int len)

{

int i = 0;

int temp = 0;

for(; i < len; i++)

{

temp = temp ^ a[i];

}

return temp;

}

int findFirstBit1(int n)

{

int count = 1;

while(!( n & 1))

{

n = n>>1;

count++;

}

return count;

}

int isBit1(int a, int count)

{

a = a >> count-1;

return (a & 1);

}

void findTwoUnique(int\* a, int len)

{

int i = 0;

int m = 0, n = 0;

int temp = findXorSum(a, len);

int count = findFirstBit1(temp);

for(; i < len; i++)

{

if(isBit1(a[i],count))

{

m = m ^ a[i];

}

else

{

n = n ^ a[i];

}

}

printf("%d, %d", m, n);

}

int main()

{

findTwoUnique(a,8);

}

18. 链表翻转。给出一个链表和一个数k，比如链表1→2→3→4→5→6，k=2，则翻转后2→1→4→3→6→5，若k=3,翻转后3→2→1→6→5→4，若k=4，翻转后4→3→2→1→5→6，用程序实现

思想：采用遍历链表，分成length/k组，对每组进行逆转，逆转的同时要将逆转后的尾和头连接起来

//#include "stdafx.h"

#include "stdio.h"

#include "stdlib.h"

#include<malloc.h>

typedef struct Node{

int value;

Node\* next;

}LinkList;

void Converse(LinkList\* pPre,LinkList\* pCur)

{ //链表逆转

LinkList\* p = NULL;

LinkList\* pNext = NULL;

p = pPre->next;

LinkList\* p1 = NULL;

if(pCur!=NULL)

pNext = pCur->next;

while( p!=pNext)

{

p1 = p->next;

p->next = pPre;

pPre = p;

p = p1;

}

}

int main()

{

int count = 0, k,i=0,j=0,flag = 1,length=0,groups = 0;

scanf("%d",&k);

LinkList\* pPre = (LinkList\*)malloc(sizeof(LinkList));

LinkList\* pCur = (LinkList\*)malloc(sizeof(LinkList));

LinkList\* pNext = (LinkList\*)malloc(sizeof(LinkList));

LinkList\* head = NULL;

LinkList\* pTempTail = NULL; //指向逆转之后的尾部

LinkList\* pTempHead = NULL;

pCur->value = 1;

pPre = pCur; //创建初始链表

for(i=2;i<=6;i++) {

LinkList\* node = (LinkList\*)malloc(sizeof(LinkList));

node->value = i;

pCur->next = node;

pCur = node;

}

pCur->next = NULL;//最后一定要置NULL，c++中用new则无须置NULL

pCur = pPre;

while(pCur!=NULL)

{

length++;

pCur = pCur->next;

}

i=0;

groups = length/k; //分成K段

pCur = pPre;

while(i<=groups)

{

count = 0;

while(count<k-1 && i<groups)

{

pCur = pCur->next;

count++;

}

if(i<groups)

{

pNext = pCur->next;

pTempHead = pCur; /\*没做翻转之前的头部,变成了翻转之后的尾部\*/

if(flag == 0)

{

pTempTail->next = pTempHead;

}

pTempTail = pPre;

Converse(pPre,pCur);

//pTempTail = pPre;

if(flag==1)

{

head = pCur;

flag = 0;

}

pCur = pNext;

}

else

{

pTempTail->next = pNext;

}

pPre = pCur;

i++;

}

pCur = head;

while(j<length) {

j++;

printf("%d",pCur->value);

pCur = pCur->next;

}

printf("\n");

// system("pause");

return 0;

}

19. 链表相邻元素翻转，如a->b->c->d->e->f-g，翻转后变为：b->a->d->c->f->e->g

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

typedef struct node{

char val;

struct node\* pNext;

}Node;

Node\* CreateList(int n);

void Traverslist(Node\* pHead);

Node\* TransNeighbor(Node\* pHead);

int main(){

Node\* pHead = CreateList(7);

printf("before transform\n");

Traverslist(pHead);

TransNeighbor(pHead);

printf("\nafter transform\n");

Traverslist(pHead);

getchar();

return 1;

}

//创建新链表

Node\* CreateList(int n){

Node\* pHead = (Node\*)malloc(sizeof(Node));

Node\* pTail = pHead;

pTail->pNext=NULL;

int i;

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

Node\* pNew = (Node\*)malloc(sizeof(Node));

pNew->val = 'a'+i;

pTail->pNext = pNew;

pNew->pNext = NULL;

pTail = pNew;

}

return pHead;

}

void Traverslist(Node\* pHead){

Node\* p = pHead->pNext;

int isFirst = 0;

while(p!= NULL)

{

if(isFirst==0)

{

printf("%c",p->val);

isFirst=1;

}else{

printf("->%c",p->val);

}

p = p->pNext;

}

return;

}

Node\* TransNeighbor(Node\* pHead){

Node\* p = pHead->pNext;

while(p->pNext!=NULL && p->pNext->pNext!=NULL)

{

char value = p->val;

p->val=p->pNext->val;

p->pNext->val=value;

p=p->pNext->pNext;

}

return pHead;

}

20. 输入一串字符串，其中有普通的字符与括号组成（包括‘（’、‘）’、‘[’,']'）,要求验证括号是否匹配，如果匹配则输出0、否则输出1.

#include<stdio.h>

#include<malloc.h>

//#define MAX 100

int main()

{

char a[100],c[]="(((1+2))";

int i=0,j=0;;

int flag=0;

while(c[i]!=NULL&&flag==0)

{

switch(c[i])

{

case('('):

case('['):

a[j++]=c[i];break;

case(')'):

if(a[j-1]=='(')

{

a[j-1]='\0';

j--;

}

else

flag=1;

break;

case(']'):

if(a[j-1]=='[')

{

a[j-1]='\0';

j--;

}

else

flag=1;

break;

}

i++;

}

if(j!=0) flag=1;

printf("%d\n",flag);

return 0;

}

方法2：#include<stdio.h>

#include<string.h>

#include <stdlib.h> // ！！！分配内存头文件

#define m 20

typedef char ElemType;

typedef struct

{

ElemType stack[m];

int top;

}stacknode;

stacknode \*sp;

Init(stacknode \*st)

{

st->top=0;

return 0;

}

void Push(stacknode \*st,ElemType x)

{

if(st->top==m)

printf("The stack is overflow!\n");

else

{

st->top=st->top+1;

st->stack[st->top]=x;

}

}

void Pop(stacknode \*st)

{

st->top=st->top-1;

}

main()

{

char s[m]="(()";

int i;

printf("Creat a stack!\n");

sp = (stacknode \*)malloc(sizeof(stacknode)); // ！！！添加的语句

Init(sp);

printf("Input a expression:\n");

// gets(s);

for(i=0;i<strlen(s);i++)

{

if(s[i]=='(')

Push(sp,s[i]);

if(s[i]==')')

Pop(sp);

}

if(sp->top==0)

printf("左右括号是匹配的！\n");

else

printf("左右括号是不匹配的！\n");

return 0;

}

２１．将第一行中含有第二行中“23”的数输出并排序

2.输入一行数字：123 423 5645 875 186523  
在输入第二行：23

将第一行中含有第二行中“23”的数输出并排序  
结果即：123 423 186523

#include<stdio.h>

#define M 20

int main()

{

int a[M];

int i,j,s,temp;

int sort[M],t=0;

char c=' ';

i=0;

while(c!='\n')

{

scanf("%d%c",&temp,&c);

a[i++]=temp;

}

scanf("%d",&s);

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

{

temp=a[j];

if(temp%100==s)

{

sort[t++]=a[j];

}

else

temp/=10;

}

for(i=0;i<t-1;i++)

for(j=0;j<t-i-1;j++)

{

if(sort[j]>sort[j+1])

{

temp=sort[j+1];

sort[j+1]=sort[j];

sort[j]=temp;

}

}

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

printf("%d ",sort[i]);

printf("\n");

return 0;

}

２２输入m个字符串 和一个整数n, 把字符串M化成以N为单位的段，不足的位数用0补齐。

如 n=8 m=9 ，

123456789划分为：12345678  
90000000

123化为 ：12300000

#include<stdio.h>

#include<string.h>

int main()

{

char c[200]={'\0'};

scanf("%s",&c);

int n,i,j;

int len=strlen(c);

scanf("%d",&n);

for(i=1;i<=len;i++)

{

j=i%n;

printf("%c",c[i-1]);

if(j==0)

printf("\n");

}

if(j!=0)

for(i=j+1;i<=n;i++)

printf("0");

return 0;

}

２３将 电话号码 one two 。。。nine zero  
翻译成1  2 。。9 0

中间会有double

例如输入：OneTwoThree  
输出：123

输入：OneTwoDoubleTwo  
输出：1222

输入：1Two2 输出：ERROR

输入：DoubleDoubleTwo 输出：ERROR

有空格，非法字符，两个Double相连，Double位于最后一个单词 都错误

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int main()

{

char a[11][11]={"zero","one","two","three","four","five","six","seven","eight","nine","double"};

char temp[11], c=' ';

int i,j,f,d=0;

while(c!='\n')

{

scanf("%s%c",&temp,&c);

f=0;

for(j=0;j<11;j++)

{

if(!strcmp(temp,a[j])&&j<10)

{

printf("%d",j);

f=1;

if(d==1)

{

printf("%d",j);

d=0;

}

}

else if(!strcmp(temp,a[j])&&j==10)

{

d=1;

f=1;

}

}

if(f==0)

break;

}

if(d==1||f==0)

printf("error\n");

printf("\n");

return 0;

}

# ２４．将整数倒序输出，剔除重复数据

输入一个整数，如12336544，或1750，然后从最后一位开始倒过来输出，最后如果是0，则不输出，输出的数字是不带重复数字的，所以上面的输出是456321和571。如果是负数，比如输入-175，输出-571。

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<malloc.h>

int main()

{

char \*input=(char\*)malloc(sizeof(char));

scanf("%s",input);

int a[10]={0},i,flag=0,flag1=0;

int len=strlen(input);

if(input[0]=='-')

{

flag=1;

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

input[i]=input[i+1];

}

int len1=strlen(input);

int n[50],temp;

int count=0;

for(i=0;i<len1;i++)

{

temp=input[i]-'0';

if(a[temp]==0)

{

n[count++]=temp;

a[temp]=1;

}

}

n[count]='\0';

if(flag==1)

printf("-");

for(int ii=count-1;ii>=0;ii--)

{

if(n[ii]!=0||flag1!=0)

{

printf("%d",n[ii]);

flag1=1;

}

}

printf("\n");

return 0;

}

２５．编程的时候，if条件里面的“(”、“)”括号经常出现不匹配的情况导致编译不过，请编写程序检测输入一行if语句中的圆括号是否匹配正确。同时输出语句中出现的左括号和右括号数量，如if((a==1)&&(b==1))是正确的，而if((a==1))&&(b==1))是错误的。注意if语句的最外面至少有一对括号。提示：用堆栈来做。

输入：if((a==1)&&(b==1))

输出：RIGTH 3 3

输入：if((a==1))&&(b==1))

输出：WRONG 3 4

#include<stdio.h>

#include<string.h>

int main()

{

char s[800]={'\0'};

scanf("%s",&s);

// char s[]="if(())";

int len=strlen(s);

int i,left=0,right=0;

int a[50],k=0,flag=1;

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

{

if(s[i]=='(')

{

left++;

a[k]=1;

k++;

}

else if(s[i]==')')

{

right++;

if(a[k-1]==1&&k>0)

{

a[k-1]=0;

k--;

}

else

flag=0;

}

if((i==2&&s[i]!='(')||(i==len-1&&s[i]!=')'))

flag=0;

}

if(a[0]==0&&flag!=0)

printf("RIGHT");

else

printf("WRONG");

printf("%d %d\n",left,right);

return 0;

}

约瑟夫问题

输入一个由随机数组成的数列（数列中每个数均是大于0的整数，长度已知），和初始计数值m。从数列首位置开始计数，计数到m后，将数列该位置数值替换计数值m，并将数列该位置数值出列，然后从下一位置从新开始计数，直到数列所有数值出列为止。如果计数到达数列尾段，则返回数列首位置继续计数。请编程实现上述计数过程，同时输出数值出列的顺序

比如：输入的随机数列为：3,1,2,4，初始计数值m=7，从数列首位置开始计数（数值3所在位置）  
第一轮计数出列数字为2，计数值更新m=2，出列后数列为3,1,4，从数值4所在位置从新开始计数  
第二轮计数出列数字为3，计数值更新m=3，出列后数列为1,4，从数值1所在位置开始计数  
第三轮计数出列数字为1，计数值更新m=1，出列后数列为4，从数值4所在位置开始计数  
最后一轮计数出列数字为4，计数过程完成。  
输出数值出列顺序为：2,3,1,4。

要求实现函数：  
void array\_iterate(int len, int input\_array[], int m, int output\_array[])

【输入】 int len：输入数列的长度；  
int intput\_array[]：输入的初始数列  
int m：初始计数值

【输出】 int output\_array[]：输出的数值出列顺序

【返回】 无

示例:  
输入：int input\_array[] = {3,1,2,4}，int len = 4， m=7  
输出：output\_array[] = {2,3,1,4}

解题思路：

每次出列一个数值，需要对m、input\_array、output\_array、输出位置outPos、起始位置startPos进行更新；

对于输出位置outPos的计算是关键！通过分析可知，outPos=(startPos+m-1)%num

代码实现：

[view plaincopy to clipboardprint?](http://blog.csdn.net/xiaobai1593/article/details/6763390)

#include <stdio.h>

void print\_array(int len, int array[])

{

for(int i=0; i<len; i++)

printf("%d ", array[i]);

printf("\n");

}

//input\_array:a[0]...a[len-1]

void array\_iterate(int len, int input\_array[], int m, int output\_array[])

{

int startPos=0;

int outPos;

int nIter=len-1;

int num=len;

for(; nIter>=0; nIter--)

{

outPos=(m+startPos-1)%num;

m=input\_array[outPos];

startPos=outPos;

printf("outPos is %d, new m is %d\n", outPos, m);

output\_array[len-nIter-1]=input\_array[outPos];

for(int i=outPos; i<num-1; i++)

input\_array[i]=input\_array[i+1];

num--;

print\_array(num, input\_array);

}

}

void main()

{

int input\_array[]={3,1,2,4};

int output\_array[4]={0};

array\_iterate(4, input\_array, 7, output\_array);

print\_array(4, output\_array);

}

27.统计数字出现的次数，最大次数的统计出来

举例：

输入：323324423343

输出：3,6

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int main()

{

char \*num="323324423343";

int a[10]={0};

int len=strlen(num);

int i,j,temp,count=0,maxnum=0;

printf("%d\n",len);

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

{

temp=num[i]-'0';

a[temp]++;

}

int temp1=a[0];

for(j=0;j<10;j++)

{

if(a[j]!=0)

{

count++;

temp1=(temp1>a[j])?temp1:a[j];

printf("%d %d\n",a[j],j);

}

}

printf("数字出现次数为:%d\n",count);

printf("最大次数为:%d\n",temp1);

return 0;

}

28. .字符串首字母转换成大写

举例：

输入：this is a book

返回：This Is A Book

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int main()

{

char input[]="this is a book";

char output[256]={'\0'};

int i,len;

len=strlen(input);

printf("变换前的字符串为:%s\n",input);

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

{

if(input[0]!=' ')

input[0]-=32;

if(input[i]==' ')

input[i+1]-=32;

output[i]=input[i];

}

printf("变换后的字符串为:%s\n",output);

}

29. 子串分离

题目描述：

通过键盘输入任意一个字符串序列，字符串可能包含多个子串，子串以空格分隔。请编写一

个程序，自动分离出各个子串，并使用’,’将其分隔，并且在最后也补充一个’,’并将子

串存储。

如果输入“abc def gh i        d”，结果将是abc,def,gh,i,d,

要求实现函数：

void DivideString(const char \*pInputStr, long lInputLen, char \*pOutputStr);

【输入】  pInputStr：  输入字符串

         lInputLen：  输入字符串长度

【输出】  pOutputStr：  输出字符串，空间已经开辟好，与输入字符串等长；

#include <stdio.h>

#include<stdlib.h>

#include<malloc.h>

#include<string.h>

void DivideString(const char \*pInputStr, long lInputLen, char \*pOutputStr)

{

int cnt;

const char \*p=pInputStr;

while(\*p!=NULL)

{

if(\*p!=' ')

{ cnt = 0;

\*pOutputStr++ = \*p++;

}

else

{ cnt++;

p++;

if(cnt==1)

\*pOutputStr++ = ',';

}

}

\*pOutputStr++ = ',';

\*pOutputStr = '\0';

}

void main()

{

char \*str = "abc def gh i d";

long len = strlen(str);

char \*outstr = (char\*)malloc(sizeof(str));

//char outstr[100];

DivideString(str,len,outstr);

printf("%s",outstr);

printf("\n");

}

9月份的华为面试没有被通知，所以参加了10月份的华为软件面试：

10月18号下午在西工大投的简历，然后通知22号机试，我利用这四天的时间对网上今年贴出的华为机试题进行了整理，并且每道题重新整理编写了一遍，有些网上贴出的答案代码对变量考虑的不周全，本人在此都进行了改正，可以保证每道题运行无错误。（以下所有机试题均不含本人所考题目，不算泄密哈…………）

23号通知了面试：9点15准时到面试点，然后是一面技术面，面试官很和蔼，首先自我介绍，然后让我找一个项目，从头到尾讲一遍，任务就是让他听懂，还有就是项目期间遇到的问题及解决方式，项目中自己的感受以及收获是什么。差不多讲了一个小时（当时面试算最长的吧），然后他又说这次面试我还没有问你软件方面的东西，LZ一听，要出问题，说，尽管我不是科班出身，但我平时一直看编程以及通信方面的书籍，我相信进入公司后，以我的学习能力……，这时面试官接过话说，我知道你的学习能力很强，然后就让我出去等通知。然后就过了，等性格测试。

纠结的性格测试，我差不多做了一个小时，建议做36分钟，但也没有准确的时间限定，当时快要吃午饭了，所以我是倒数第二批进去测试的，但确是最后一个出来的……，差不多10秒吧，通知我先去吃午饭，（华为的午餐还是不错的。服务态度也很好）。

最后就是下午的二面，（终面）了。终面不是传说中的ＨＲ，还是研发的，感觉像研发部门的经理，这点让网上介绍骗了……，还是自我介绍，再是介绍一个具体项目。然后就是问我有啥问题想问他的，开始我问了个：您对我这一会的面试的感觉是怎样的。他笑了下说怎么都是这个问题，面试期间不方便回答面试相关的问题，让我再重新问了一个，于是就问了一个，网上都说华为工作非常累，非常忙，真实情况是如何的（后来感觉这个问题非常棒＾＾，勾起了面试官的很多美好回忆，跟我扯了差不多１５分钟），然后我又引到我对加班以及工作的忙与累的看法上，以及对进入华为的愿景。然后就让回去等通知了。欣喜的是２４号下午收到通知，２５号签约，早上去了，果断签了

本人本科是一普通二本，物理专业出身，读研转为目标自主识别，华为一直是我工作中最佳的一个选择，从９月份到现在的１０月底，我也拿到了四五个ｏｆｆｅｒ，但都拒了，一直等华为，呵呵现在如愿了。我自认能力不出众，大的软件公司勉强通过笔试，都死在了面试上，特别是面阿里的时候，面试官重光老师（ｉ应该这样写呵呵），一针见血的指出了我的不足，并且给我了很多非常中肯的建议，在我以后的找工作中起到了非常大的帮助，在此非常感谢他。就写这些吧。