

1.1. 设 a 为 7，则执行下列语句后，写出 b 的值不为 3 的语句。 C

A) $b = a/2;$

B) $b = 9 - (--a);$

C) $b = a\%2;$

D) $b = a < 3 ? 2 : 3;$

1.3. 设有定义语句: B

```
struct  
{  
    int x;  
    int y;  
} d[2]={{1,3},{2,7}};
```

则 $d[1].x + d[0].y - d[0].x$ 的值是多少?

A) 3

B) 4

C) 5

D) 6

1.6. 循环语句 `for(x=0,y=0;(y!=123)||x<4;x++);`的循环次数是多少: **D**

- A) 不确定
- B) 4 次
- C) 3 次
- D) 无限次

1.9. 执行以下程序代码:

```
#define M(a,b) (a)>(b)? (a):(b)
main() {int i=10,j=15; printf(“%d\n”,10*M(i,j));}
```

输出的结果是 **C**

- A) 100
- B) 150
- C) 10

1.10. 函数调用语句 f((1,2),(1,2,3))中函数 f 的参数个数是: **A**

A) 2

B) 3

C) 4

D) 5

2.1.请写出以下程序的运行结果。 **0,7,0**

```
#include <stdio.h>
void main()
{
    int a=10,b=4,c=3;
    if(a<b) a=b;
    if(a>c) a=c;
    printf(“%d,%d,%d \n”, a/b, a|b, a&b);
}
```

4.1. (9分) 插入排序是一种基础的排序算法。其基本思想如下：设待排序的数组 a 中 n 个节点的序列为 a_0, a_1, \dots, a_{n-1} 。按照节点顺序处理 a_i 时, a_0, a_1, \dots, a_{i-1} 已经排序, 则将 a_i 和 a_{i-1}, a_{i-2}, \dots 依次进行比较, 将比 a_i 大的向右移动, 直到发现比 a_i 小的元素或是到头为止, 此时可以确定 a_i 应放置的位置。完成以下 `insert_sort` 函数, 以实现插入排序。

```
void insert_sort(int a[],int n)
{
    int i,j,t;
    for(i=1;i<n;i++)
    {
        t = a[i];
        for(j=i-1; 4.1.1 ; j--) (j>=0&& t<a[j])
            4.1.2 ; a[j+1]=a[j];
        4.1.3 ; a[j+1]=t;
    }
}
```

4.2. (9分) 读入一个不超过6位的正整数。要求求出各位数字之和，并求出它是几位数且打印。

例如读入数为3210，应该输出以下格式的信息：

各位数字之和是：6，数据位数：4

```
#include<stdio.h>
void main()
{
    long a;          /* 不超过6位的正整数 */
    int bit ;       /* 数据a的位数 */
    int sum;        /* (当前) 数字之和 */
    printf("请输入一个不超过6位的正整数:");
    scanf("%ld", &a);
    printf("各位数字之和是: ");
    4.2.1 ;          //bit = sum =0;
    while(a)
    {
        sum += 4.2.2 ; // a % 10
        4.2.3 ;      // a /= 10 ;
        bit++;
    }
    printf(" %d, 数据位数: %d\n", sum, bit);
}
```

5.1. (10 分) 编写函数 `strsub(char *s1, int i, int j)`, 从串 `s1` 的位置 `i` 开始取长度为 `j` 的子串。要求 (1) 申请新的内存空间存放该子串, `strsub` 返回新串的指针; (2) 对参数 `i` 和 `j` 的有效性进行必要的检查; 如失败, 返回 `NULL`; (3) 不可以使用字符串的标准库函数。

```
#include "stdlib.h"
char * strsub(char *s1, int i, int j)
// 从串 s1 位置 i 开始取长度为 j 的字串返回
{ // 参考答案
    int length = 0;
    for (; s1[length]&& i+j>length; length++);
    if(i<0 || j<0 || i>=length || i+j>length)
        return NULL;
    char *s = (char *)malloc(sizeof(char)*(j+1));
    for(int temp=0; temp<j; temp++)
        s[temp]=s1[i+temp];
    s[j]='\0';
    return s;
}
```

5.2 (10 分) N 位超级质数。从标准输入读入一个整数 $N(N < 9)$, 输出所有满足以下条件的 N 位超级质数: 左侧前任意连续位均是质数。例如, 23, 233, 2339, 23399 就分别是这样的 2 位、3 位、4 位、5 位超级质数。提示: 首先以所有的一位质数 (即 {2, 3, 5, 7}) 为种子集合, 然后在每个种子的后面增加一个数字 (只可能来自于 {1, 3, 7, 9}), 并检查新生成的两位数是否是质数。如果新生成的两位数依然是质数, 则也把它加到种子集合中。重复这一过程, 就可以从种子集合中生成所有长度为 N 的超级质数。假定已有函数 `isprime`, 它的参数为整数 x 。若 x 为质数, 则 `isprime` 返回 1, 否则返回 0。

```
#define numof(a) (sizeof(a)/sizeof(a[0]))
int hd = 0, tail =4, queue[Max]={2,3,5,7}; //假定 Max 足够大,queue 存放种子
int digits[]={1,3,7,9}; //可能的新尾数
```

```
void main()
{
    int i,m,n,len,end = 0;
    scanf("%d",&n);
    for(len=1;len<n;len++)
    {
        for(end=tail;hd<end;hd++)
        {
            m=queue[hd]*10;
            for(i=0;i<numof(digits);i++)
                if(isprime(m+digits[i]))
                    queue[tail++] = m+digits[i];
        }
    }
    for(i=end;i<tail;i++)
        printf("%d\n",queue[i]);
}
```