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;

1.2. 以下对于 c 语言函数的有关描述中，正确的是。 C

A）C 函数既可以嵌套定义，也可以递归调用

B）函数必须有返回值

C）调用函数时，只能把实参的值传递给形参，形参的值不能传递给实参

D）函数只能调用定义在它前面的函数

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

1.4. 设 int x[]={2,3,4,5,6}, *p=x;　则不能正确引用数组元素的表达式是：A

A）*(--p)

B）*(p++)

C）*(p+2)

D）*(p--)

1.5. 下列语句中，能正确进行字符串赋值操作的是：　D

A）char *s; scanf("s%",s);

B）char s[5]={'a','b','c','d','e'};

C）char s[5]={"abcde"};

D）char *s = "abcde";

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

1.7. 在 16 位机器上，存储整形数据-2222 时，在二进制文件和正文文件中分别占用的字节数是：A

A）2,5

B）2,2

C）5,2

D）5,5

1.8. 设有定义 int a=5,b,*p=&a;则下列语句中不能使 b 的值为 5 的语句是： D

A）b=*&a;

B）b=a;

C）b=*p;

D）b=*a;

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

D）15

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);
}
```

2.2 请写出以下程序的运行结果。 62

```
#include <stdio.h>
void main()
{
     int y = 9;
     for(;y>0;y--)
       if(y%2 == 0)
          {
              printf("%d", y--=2);
              continue;
          }
 }
```

2.3 请写出以下程序的运行结果。 1089 1

```
#include <stdio.h>
void main()
{
    int a=0XAAF6, mask=0XFF32, bit=2, b;
    b = (a & mask) >> bit;
    while(b)
      {
        if(b % 2)
            printf("%d  ", b);
        b/=10;
      }
    printf("\n");
}
```

2.4.假定在当前目录下有两个文本文件，其名称和内容如下： 234252627

文件名：a1.txt a2.txt

内容： 121314# 252627#

假定所有文件操作函数都可以成功执行，则以下程序的输出是：

```
#include <stdio.h>
void fc(FILE *fp1)
{
    char c;
    while((c=fgetc(fp1))!='#')
    {
        if(c>'1')
          putchar(c);
    }
}
void main()
{
  FILE *fp = fopen("a1.txt","r");
  fc(fp); fclose(fp);
  fp = fopen("a2.txt","r");
  fc(fp); fclose(fp);
}
```

2.5.请写出以下程序的运行结果。 4

```
#include <stdio.h>
int sub(int n)
{
    static int y;
    if(n > 3)
        y += sub(--n);
    return y+1;
 }
void main()
{
    int x;
    x = sub(5);
    printf("%d\n", x);
}
```

3.1.（9 分）以下程序将已知数字字符序列转换成十进制整数，并输出结果。请在不删减语句的情况下指明程序中的第几行有错误，并且写出正确的语句。

```
/*第 1 行*/        #include <stdio.h>
/*第 2 行*/        void main()
                   {
/第 3 行/                char num[10] = 31725;   // "31725"
/第 4 行/                int i, n = 0;
/第 5 行/                for(i=0; num[i]>=0&&num[i]<=9; i++)
// i<sizeof(num)/sizeof(num[0])&&num[i]>='0'&&num[i]<='9'
/第 6 行/                    n = n * 10 + num[i]; //n = n * 10 + num[i]–'0';
/第 7 行/                printf(“%d\n”, n);
                   }
```

3.2.（6 分）以下程序运行的结果应该为"a=5, b=3"，请在不增删语句的情况下指明程序中的第几行有错误，并且写出正确的语句。

```
/* 第 1 行 */ void swap(short *x, short *y)
                   {
/* 第 2 行 */          short *z;    // short z;
/* 第 3 行 */          z = *x;
/* 第 4 行 */          *x = *y;
/* 第 5 行 */          *y = z;
                   }

                   void main()
                   {
/* 第 6 行 */          short a=3, b=5;
/* 第 7 行 */          swap(a, b);    // swap(&a,&b);
/* 第 8 行 */          printf("a=%d, b=%d\n", a, b);
                   }
```

请在以下试题中写出应填入_____处的字句，以完成各题程序所需要的功能。

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，...，a_{n-1}。按照节点顺序处理 a_i 时，a_0，a_1，...，a_{i-1} 已经排序，则将 a_i 和 a_{i-1}，a_{i-2}，...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);
}
```

4.3. （12 分）给定一个单链表，链表中节点的定义如下：

```
struct node
{
    int   num;
    struct node *next;
};
```

编写函数 list_rev 实现该单链表的逆置。list_rev 以要逆置的单链表头指针（指向链表的首节点）为参数，并返回逆置后的链表的新头指针。list_rev 基于递归函数的思想，分成三步：（1）逆置原始链表首节点的后续链表；（2）将原首节点接到逆置后的后续链表的末尾(即原首节点的后一个节点)；（3）将原首节点的后续置为 NULL。递归过程的终止条件是链表为空，或者只有一个节点。根据这些说明，补充完整下面的程序：

```
struct node * list_rev(struct node *h)
{
    struct  node *p;
    if(h==NULL||h->next==NULL)
      return h;
    p=   4.3.1   ;    list_rev(h->next);
     4.3.2   = h;     h->next->next;
    h->next =   4.3.3;     NULL
    return   4.3.4   ;     p
}
```

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]);
      }
```