主题作业三：函数与结构

一．单选题

1. 针对如下定义，合法的表达式是__C___。

```
struct node {
    char s[10];
    int k;
} p[5];
```

A．p.k=2　　B．p[0]->k=2　　C．(p->s)[0]='a'　　D． p[0].s="a"

数组变量名也是指针
访问结构指针所指向的结构的成员变量时用->，访问结构的成员变量时用.
成员 s 是数组，不能直接赋值（但是初始化数组的时候可以，例如 char s[10]="abc"；那是由编译器在编译阶段完成的）

2. 针对如下定义：

```
static struct {
    int x, y[3];
} a[3] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}}, *p;
p = a+1;
```

表达式*((int *)(p+1)+2)的值为__D___。

A．3　　B．7　　C．10　　D． 11

p 指向 a[1]；然后 p+1 指向 a[2]，即元素 9 的位置；
然后(int*)(p+1)则是将指针强制转换为一个整数指针，位置不动，还是 9 的位置；
+2 操作则把上面的整数指针向后移动 2 个，即指向 11 的位置；

3. 对于以下结构定义，++p->str 中的++加在__A___。

```
struct {
    int len;
    char *str;
} *p;
```

A、指针 str 上　　B、指针 p 上　　C、str 指的内容上　　D、以上均不是

->的优先级高于++，该表达式等价于++（p->str）
所以，++作用于 p 的成员变量 str 上。
执行后的结果是：指针 p 不变，但是它指向的结构的成员 str 加 1

4. 根据声明 int (*p)[10], p 是一个__A___。

A．指针　　B．数组　　C．函数　　D．数组元素

5. 若下面程序中所有的变量均已声明或定义，则下列选项中的变量能够在 fun()中使用的是__A___。

```
#include <stdio.h>
void fun(int x)
{
    static int y;
    ……
    return;
}
int z;
void main( )
{
    int a,b;
    fun(a);
    ……
}
```

A．x, y　　B．x, y, z　　C． a,b,y,z　　D．a,b,x,y,z

第 4 题解答：注意：int (*p)[10] 和 int *p[10]是完全不一样的
int *p[10]定义了一个数组，其元素类型为 int*；此时 sizeof(p)为 40。
int (*p)[10]定义了一个指针，其类型是长度为 10 的整型数组；此时 sizeof(p)为 4。

变量遵循先声明再引用的原则
x 作为函数 fun 的形式参数变量，已经声明为 int 型，可以使用
y 是函数 fun 的局部静态变量，也已经声明，也可以使用
z 在 fun 之后定义，之前没有声明，不能使用
a,b 则是函数 main 的局部变量，fun 不能使用。

注意：这里的先后顺序只得是编译器编译语句的顺序，不是语句运行的先后顺序。因为 if/switch 分支、for/while 循环等流程控制语句可能动态的控制程序语句的执行次序，所以语句的执行次序不是固定的。

二．　填空题

6. 对于下面的定义，(s[0].b)/(++p)->a 的值为__0___。

```
struct {
    int a;
    int b;
} s[2]={2,4,6,8}, *p=s;
```

s[0] 为{2，4}，s[1] 为{6，8}。
s[0].b 为 4
p 初始指向 s[0]，++p 指向 s[1]，故（++p）->a 的值为 6
因为 4/6 为 0，所以。。。。

7. 下列程序段执行后，z 的值是 _8_。

```
static struct {
    int x, y[3];
} a[3] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}}, *p=a+3;
int z;
z=*((int *)(p-1)-1);
```

结构数组 a[3]在内存中也可以看做是一个长度为 12 的整数数组：1，2，3，…， 11，12。
p 的初始地址为 a+3，那么 p-1 指向 a[2]，即元素 9 的位置（int*）将 p-1 强制转换为一个整型指针，但是还是指向 9 的位置。所以-1 作用在整型指针(int*)(p-1)上，得到 9 的前面一个整数元素的地址，即 8 的地址。

8. 下列代码段将会打印出__COND___。

```
char *c[3]={“FIRST”, “SECOND”, “THIRD”};
printf(“%s”, *(c+1)+2);
```

c 是一个指针数组，c[0]指向字符串”FIRST”,c[1]指向字符串“SECOND”，c[2]指向字符床“THIRD”。
(c+1)为 c[1]，所以 i(c+1)+2 指向字符 SECOND 中的 C.

参考第 8 题的解释

9. 下列程序段的输出结果是_fgh____。

```
char  *st[ ]={"abcd","efgh","ijkl","mnop"};
printf("%s", *(st+1)+1);
```

参考第 8 题的解释。*(st+2)等价于 st[2]。
因为*优先级高于+，所以**st+1 等价于 st[0][0]+1 即，’O’+1，即’P’

10. 下列代码段的输出为_FOUR, P____。

```
char  *st[ ]={"ONE","TWO","FOUR","K"};
printf("%s, %c\n", *(st+2), **st+1);
```

P 指向 a+1，所以*p 为 a[1]。**p 为 a[1]所指向的字符串的第一个字符，即’4’。所以**p-1 为’3’

11. 下列代码段的输出__45,3___。

```
char *a[]={“678”,”45”},**p=a+1;
printf(“%s,%c”,*p,**p-1);
```

a[0]指向第 0 行。a[1]指向第 1 行，即元素 3 的位置。
那么 a[1]+1 指向元素 3 的后面的一个位置，即 4 的位置。

12. 对于数组 a，*(a[1]+1)的值为__4___。

```
int a[3][2]={1,2,3,4,5,6};
```

13. 对于以下递归函数 f，调用 f (3) 的返回值是__-17___。

```
f (int n)
{   return ((n>0) ? 2*f(n-1)+f(n-2) : -1);  }
```

f(-1)	f(0)	f(1)	f(2)	f(3)
-1	-1	-3	-7	-17

14. 下列代码段将会打印出__4___。

```
int i;
int f(int x)
{   static int k = 0;
    x+=k++;
    return x;
}
i=f(2);
i=f(3);
printf(“%d”,i);
```

初始化：k 为 0，
调用 f(2)：k 从 0 变到 1，函数返回 2+0=2
调用 f(3)：k 从 1 变到 2，函数返回 3+1=4

15. 下列代码段将会打印出___18__。

```
int f(int x)
{   return ((x>0)? x*f(x-1):3); }
printf(“%d”,f(f(1)));
```

f(0)	f(1)	f(2)	f(3)
3	3	6	18

f(f(1)) = f(3) = 18

16. 用 typedef 写出类型定义__typedef int * AIP[10];____，使得 AIP 表示含有 10 个元素的整型指针数组类型。

typedef TOld TNew[10];
定义了一个新的类型，名称为 TNew，它是一个长度为 10 的数组，其中每一个元素类型为 Told。对于该题，Told 为 int*，TNew 为 AIP。

而 typedef TOld (*PNew)[10]; 则定义了一个新的类型的指针。该语句等价于：
typedef TOld TNew[10];
typedef TNew* PNew;

17. 假定可执行程序文件名字为 prog，则运行命令：prog hello world 后的输出为_____。

```
#include <stdio.h>
int main(int argc, char *argv[]) {
    printf("%s#%s#", argv[0], argv[argc - 1]);
    return 0;
}
```

输出为：prog#world#

argc 为 3，
argv[0],[1],[2]分别为 “prog”, “hello”和”world”

18. 下列程序段的输出是___-3_____。

```
#define FB(a,b)   (a*b+1)
int k=3;
k= FB(k+1,k-1)-9;
printf("%d",k);
```

展开宏 FB(k+1,k-1)后得到：
（k+1*k-1+1）
所以该语句为：k=（k+1*k-1+1）-9 = -3

三、 阅读理解题

19. 下列程序的输出为_6#18#____。

```
#include <stdio.h>
main()
{   int s=0,i,a[3][3]={1,2,3,4,5,6,7,8,9};
    for(i=0; i<3; i++)
        s+= *(*a+i);
    printf("%d#", s);
    for(i=0; i<3; i++)
        s+= **(a+i);
    printf("%d#", s);
}
```

s += a[0][i]

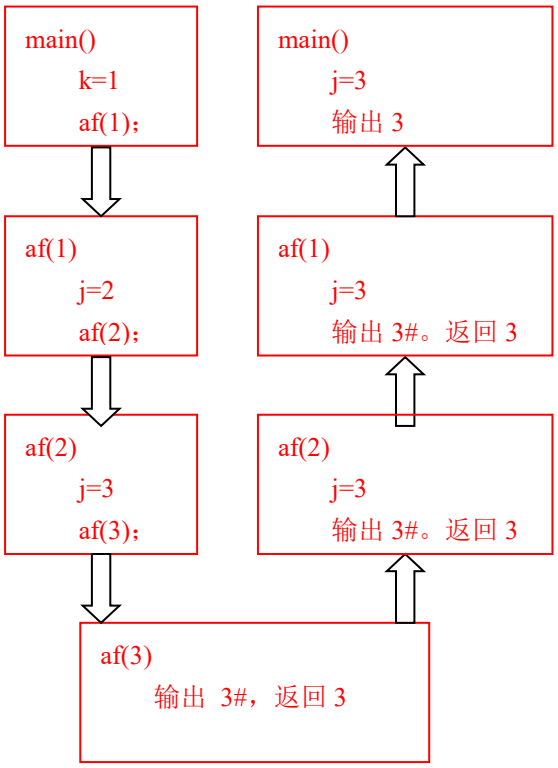
s += a[i][0]

a[3][3]的元素排列为：
1 2 3
4 5 6
7 8 9
*(*a+i) 等价于*(a[0]+i)等价于 a[0][i]。
所以第一个 for 循环后，s 为 1+2+3=6
**(a+i)等价于*a[i]等价于 a[i][0];
所以第二个 for 循环后，s 为 6+1+4+7=18

20. 下列程序的输出是__5___。

```
int f (int x)
{
    if(x<=1) return 1;
    else return f(x-1)+f(x-2);
}
void main( )
{   printf("%d", f(4));
}
```

f(0)	f(1)	f(2)	f(3)	f(4)	f(5)
1	1	2	3	5	7



21. 下列程序的输出是__3#3#3#3___。

```
#include <stdio.h>
int main (void)
{   int k = 1;
    int a_function ( int   j );
    k = a_function ( k );
    printf ( "%d" , k );
}
int a_function ( int   j )
{
    if ( j < 3 ) {
```

```
            j++;
            j = a_function ( j );
        }
        printf ( "%d#" , j );
        return ( j );
    }
```

22. 下列程序的输出为__a = 1, b = 4 ___。

```
#include   <stdio.h>
void melon (int g, int * h);
int main (void)
{     int a = 1, b = 2;
      melon ( a, &b );
      printf ( "a = %d, b = %d",   a, b );
}
void melon (int b, int * c)
{     b++;
      *c = *c + b;
}
```

melon(a, &b)执行后，a 的值不变，
b 的值增加了 a+1，即 b 变为 4。

23. 下列程序的输出为_225____。

```
#include <stdio.h>
#include<math.h>
int prime(int n)
{
   int i,m;
   if(n==1) return 0;
   m=sqrt(n);
   for(i=2;i<=m;i++)
      if(n%i==0) break;
   return i>m;
}
void main()
{   int num,i;
    num=20;
    for(i=2;i<=num;i++){
        while(prime(i)&&(num%i==0)){
            printf("%d ",i);
            num/=i;
        }
    }
}
```

Prime(n) 判断 n 是否为素数，是返回 1，否则返回 0

计算并输出 num 所有的素数因子
num =20, i=2, print 2
num =10,i=2, print 2
num =5,i=2 跳出 while
num =5,i=3,
num =5,i=4,
num =5,i=5,print 5
num =1,i=5,跳出循环

24. 下列程序的输出为__1，2___。

```
#include <stdio.h>
void f(int *x,int *y)
{   int *p;
    p=x; x=y; y=p;
```

函数 f 里交换指针，
但是交换的结果并不能返回给调用者，
也就是说，在调用者看来，都是无用功，等于啥也没做。

```
}
void main()
{   int x=1, y=2;
    f(&y, &x);
    printf("%d, %d", x, y);
    }
```

25. 若输入 ***this is a test.<ENTER>***，则下列程序的输出_This Is A Test.____。

```
#include <stdio.h>
#define TRUE 1
#define FALSE 0
int change(char *c,int status);
void main()
{
    int flag=TRUE;
    char ch;
    do{
        ch=getchar();
        flag=change(&ch,flag);
        putchar(ch);
    } while(ch!='.');
    printf("\n");
}
int change(char *c,int status)
{
    if(*c==' ') return TRUE;
    if(status&&*c<='z'&&*c>='a') *c+='A'-'a';
    return FALSE;
}
```

Status 的意义是：是否要求变成大写字母

碰到空格了，后面的第一个字符要求大写
也就是单词的第一个字母大写

26. 下列程序的输出为__5#0#___。

```
# include <stdio.h>
int f(int m)
{
    static int k=0;
    int s=0;
    for(; k<=m; k++) s++;
    return s;
}
void main( )
{
    int s=1;
    s=f(4);
    printf("%d#%d#", s, f(2));
}
```

执行 f(4)时，m=4。执行后，static int k 变成了 k=5，函数返回 5。
所以 main 中 s=5。
再执行 f(2)时， m=2，k=5，所以 f 返回 0。
所以输出为 5#0#

27. 下列程序的输出为_-1#-1#-2#____。

```
#include <stdio.h>
```

```
int fun(int x)
{
    int t;
    if(x<=0)
        t=x;
    else
        t=fun(x-1)+fun(x-2);
    return t;
}
void main()
{
    int i;
    for(i=1;i<=3;i++)
        printf("%d#", fun(i));
}
```

f(-1),	f(0),	f(1)	f(2)	f(3)
-1	0	-1	-1	-2

28. 下列程序将会打印出__9#1#___。

```
#include <stdio.h>
#define my_square_add(a, b) (a * a + b * b)
#define my_square_sub(a, b) (a * a - b * b)
int main()
{
    int a = 1, b = 2;
    printf("%d#", my_square_add(a + b, b));
    printf("%d#", my_square_sub(a + b, b));
    return 0;
}
```

宏定义展开
msadd(a+b,b) ➔ a+b*a+b+b*b = 9
mssub(a+b,b) ➔ a+b*a+b-b*b = 1

29. 下列程序的输出结果是_1#3#5#____。

```
# include <stdio.h>
int f(int x)
{   static int k=0;
    return ++k+x;
}
main( )
{  int k;
   for(k=0;k<3;k++)   printf("%d#", f(k) );
}
```

执行 f(0)后：static k=1，返回 1+0=1
执行 f(1)后：static k=2，返回 2+1=3
执行 f(2)后：static k=3，返回 3+2=5

30. 下列命令行参数程序生成的执行程序为 test.exe，执行 ***test 123 abc<回车>***，输出结果是_abc#123#____。

```
# include <stdio.h>
main(int argc,char *argv[])
{
    while(--argc)
        printf("%s#",argv[argc]);
}
```

while 的条件表达式为--argc，所以当 argc 为 3、2 时，该条件成立。条件成立时，分别输出 argv[2] 和 argv[1]。因此答案为。。。。

argc 为 3
argv[0]为 test
argv[1]为 123
argv[2]为 abc

31. 假设有下列函数定义：

```
void foo(int sum)
{
   int j;
   for (j = 0; j < 10; ++j)
      sum += (j + 1) * sum;
}
```

在main函数中有如下代码段：

```
int sum = 0;
foo(sum);
printf("%d", sum);
```

执行后，输出结果为___0_____。

Main 函数调用 foo(sum)之后，sum 的值不会被 foo 函数改变。