程序设计期末复习

1. 请写出以下程序的运行结果。

```
struct
{
    int a, b;
} d[2] = {{2, 8}, {5, 9}};
printf("%d\n", d[0].a + d[1].b++ - --d[1].a);
```

A) 8　　B) 6　　C) 7　　D) 10

2. 请指出算法和数据结构不适合的一种情况。

A) 顺序查找算法和数组

B) 二分查找算法和链表

C) 顺序查找算法和链表

D) 冒泡排序算法和数组

3. 与 a[i][j]的表达不等价的是（　　）。

A) *(a+i+j)　　B) *(a[i]+j)

C) *(*(a+i)+j)　　D) (*(a+i))[j]

4. 有以下变量定义，则后续正确的语句是：（ ）

```
int a[3], *p, i = 2, j ;
```

A） a++;　　B） a = {1,2,3};　　C） p = &(i + j);　　D） p = a+i;

B）数组的初始化只能在定义时完成，不能直接在赋值语句中整体赋值。
C）i ＋ j是一个整数，不能对其取地址，取地址操作只能用于变量。

5. 针对以下定义，错误的语句是：（ ）

```
char str[] = "hello", *p = "hi";
```

A） p = str;　　B） str[5] = 'a';　　C） p[0]= 'a';　　D） p = &str[3];

指针指向字符串字面量，不可修改，但是用数组的话会复制到数组的每一位

6. 文本文件 test.txt 中原来保存的内容为：123。以只写模式"w"打开该文件，文件指针保存在变量 fp 中。运行语句“fprintf(fp, "456");”后关闭文件，test.txt 中的内容为（ ）。

A) 456　　B) 123　　C) 456123　　D) 123456

只写模式是清空内容的

7. 对于定义在某个函数中的局部自动变量，以下说法错误的是 ()

A) 比全局变量生命期短，因此节省内存　　B) 相关内存空间的释放无需程序员管理

C) 可以使用各种基础类型，也包括数组和结构　　D) 通过将变量地址赋值给全局指针变量，可以在任意其他函数中使用

8. 以下程序的输出结果为 ()

```
#include <stdio.h>

int v1 = 10;

void function1() {

    int v1 = 20;

}

int main() {

    int v2 = 40;

    printf("%d ", v2);

    printf("%d ", v1);

    function1();

    if (v1 == 20) {

        int v2 = 50;

        printf("%d\n", v2);

    }

    return 0;

}
```

定义在函数内部的局部变量和全局变量同名时，全局变量会暂时被隐藏

A) 40 10　　B) 40 10 50　　C) 40 10 40　　D) 编译错误

9. 有如下定义：

```
struct A{
        int arr[2];
} ins = {{2,3}};
```

考虑四种实参形式 (1) &ins (2) ins.arr (3) ins (4)ins.arr[0], ins.arr[1]，其中可能用于编写函数来交换 ins.arr[0]值和 ins.arr[1]值的形式是：

A）只有（1）　　B）只有(1)(2)　　C）只有(1)(2)(3)　　D）都可以

传入地址和指针可以修改值，直接传值不行

10. 下面程序段的输出结果为：

```
#include <stdio.h>

#define SQUARE(x) x * x

#define XOR(x) x ^ x

int main() {

    int num = 5;

    int result1 = SQUARE(num + 1);

    int result2 = XOR(num + 1);

    printf("%d,  %d\n", result1, result2);

    return 0;

}
```

结果为11， 0

宏是字符串替代

请写出以下程序的运行结果。

```
#include "stdio.h"

int main(){

    int arr[][5] = {{1,2},{3,4,5},{6,7,8,9},{10,11,12,13,14}};

    int *p = &arr[0][0];

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

        printf("%d    ",*(p+i*i));

    return 0;

}
```

没有初始化自动填充为0

1 2 0 0 11

2. 请写出以下程序的运行结果。

```
#include <stdio.h>

int main() {

    int arr[5] = {1, 2, 3, 4, 5}, start = 0, end = 3;

    while (start < end) {

        int temp = arr[start];

        arr[start] = arr[end];

        arr[end] = temp;

        start++;        end--;

    }

    for (int i = 0; i < 5; i++)    printf("%d ", arr[i]);

    return 0;

}
```

4 3 2 1 5

3.设有链表序列 10->5->15->7->12,已知指针 head 指向链表头，请写出函数 f 的功能及

f(head, 8)的返回值。

```
typedef struct Node {
    int data;
    struct Node* next;
} Node;
```

```
int f (Node* head, int k) {
    int s1 = INT_MAX, s2 = INT_MAX; // INT_MAX 为最大整数
    Node* current = head;
    while (current != NULL) {
        if (current->data > k) {
            if (current->data < s1) {
                s2 = s1;
                s1 = current->data;
            } else if (current->data < s2 && current->data != s1) {
                s2 = current->data;
            }
        }
        current = current->next;
    }
    return s2;
}
```

比k大的所有数字中，第二小的数字

12

4. 出以下程序的运行结果。

```
#include <stdio.h>
#include <string.h>
void swap(char* a, char* b) {
    char temp = *a;   *a = *b; *b = temp;
}
void func(char* str, int start, int end) {
    if (start == end)
        printf("%s\t", str);
    else {
        for (int i = start; i <= end; i++) {
            swap(&str[start], &str[i]);
            func(str, start + 1, end);
            swap(&str[start], &str[i]);
        }
    }
}
int main() {
    char str[] = "abc";
    int len = strlen(str);
    func(str, 0, len - 1);
    return 0;
}
```

递归找到字符串的全排列

abc—>bc　　bc—>c　　cb

bac—>ac　　cba—>ab

abc　acb　　bac　bca　　cba　　cab

1. 以下函数计算两个以字符串表示的整数的和。例如当 num1= "1234"， num2 = "9876" 时，result[] = "11110"。

```
void add(char* num1, char* num2, char* result) {

    int len1 = strlen(num1);

    int len2 = strlen(num2);

    int carry = 0;  // 进位标志

    int i=len1-1, j=len2-1, k=0; //加法从低位到高位；k 是 result 下标

/*1*/   while (i>=0 && j>=0 && carry>0) { // i>=0 || j>=0 || carry>0

/*2*/       int digit1 = i >= 0 ? num1[i] - '0' : 0;

/*3*/       int digit2 = j >= 0 ? num2[j] - '0' : 0;

/*4*/       int sum = digit1 + digit2 + carry;

/*5*/       result[k]= sum % 10;  // result[k] = (sum % 10) + '0';

/*6*/       carry = sum / 10;

/*7*/       k++, i--, j--;

          }

/*8*/   result[k] = '\0';

/*9*/   int len = strlen(result);

/*10*/  for (i = 0; i < len; i++) { // i = 0; i < len / 2; i++//for 循环用于颠倒 result 中的数字，得到最终结果

            char temp = result[i];

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

            result[len - i - 1] = temp;

        }

    }
```

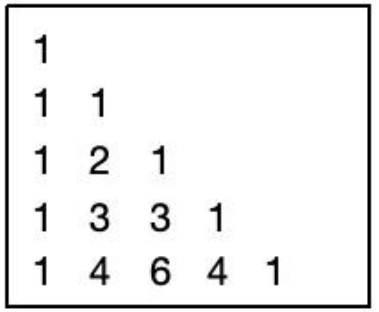
以下函数计算两个以字符串表示的整数的和。例如当num1= "1234"， num2 = "9876"时，result[] = "11110"。

```
    void add(char* num1, char* num2, char* result) {
            int len1 = strlen(num1);
            int len2 = strlen(num2);
            int carry = 0;  // 进位标志
            int i=len1-1, j=len2-1, k=0; //加法从低位到高位；k是result下标
  /*1*/ while (i>=0 && j>=0 && carry>0) {
  /*2*/ int digit1 = i >= 0 ? num1[i] - '0' : 0;
  /*3*/ int digit2 = j >= 0 ? num2[j] - '0' : 0;
  /*4*/ int sum = digit1 + digit2 + carry;
  /*5*/ result[k]= sum % 10;
  /*6*/ carry = sum / 10;
  /*7*/ k++, i--, j--;
            }
  /*8*/ result[k] = '\0';
  /*9*/ int len = strlen(result);
  /*10*/ for (i = 0; i < len; i++) {//for循环用于颠倒result中的数字，得到最终结果
            temp = result[i];
            result[i] = result[len - i - 1];
            result[len - i - 1] = temp;
        }
    }
```

以下函数计算两个以字符串表示的整数的和。例如当num1= "1234"， num2 = "9876"时，result[] = "11110"。

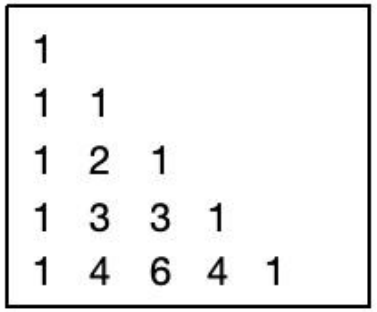
```
    void add(char* num1, char* num2, char* result) {
            int len1 = strlen(num1);
            int len2 = strlen(num2);
            int carry = 0;  // 进位标志
            int i=len1-1, j=len2-1, k=0; //加法从低位到高位；k是result下标
  /*1*/ while (i>=0 && j>=0 && carry>0) { // i>=0 || j>=0 || carry>0
  /*2*/ int digit1 = i >= 0 ? num1[i] - '0' : 0;
  /*3*/ int digit2 = j >= 0 ? num2[j] - '0' : 0;
  /*4*/ int sum = digit1 + digit2 + carry;
  /*5*/ result[k]= sum % 10;  // result[k] = (sum % 10) + '0';
  /*6*/ carry = sum / 10;
  /*7*/ k++, i--, j--;
            }
  /*8*/ result[k] = '\0';
  /*9*/ int len = strlen(result);
  /*10*/ for (i = 0; i < len; i++) { // i = 0; i < len / 2; i++//for循环用于颠倒result中的数字，得到最终结果
                temp = result[i];
                result[i] = result[len - i - 1];
                result[len - i - 1] = temp;
        }
    }
```

计算杨辉三角形的函数。pascalTriangle(5)的结果如右图。




```
   void pascalTriangle(int n) {//假设n小于10
          int triangle[10][10]; //存放结果的矩阵
   /*11*/ for (int i = 0; i < n; i++) {
   /*12*/     for (int j = 0; j <= i; j++) {
   /*13*/         if (j == 0)
   /*14*/             triangle[i][j] = 1;
   /*15*/         else
   /*16*/          triangle[i][j] = triangle[i-1][j] + triangle[i][j-1];
   /*17*/         printf("%d ", triangle[i][j]);
              }
   /*18*/     printf("\n");
         }
 }
```

计算杨辉三角形的函数。pascalTriangle(5)的结果如右图。




```
   void pascalTriangle(int n) {//假设n小于10
          int triangle[10][10]; //存放结果的矩阵
   /*11*/ for (int i = 0; i < n; i++) {
   /*12*/     for (int j = 0; j <= i; j++) {
   /*13*/         if (j == 0)  // j == 0 || j == i
   /*14*/             triangle[i][j] = 1;
   /*15*/         else
   /*16*/          triangle[i][j] = triangle[i-1][j] + triangle[i][j-1];
            // triangle[i][j] = triangle[i-1][j-1] + triangle[i-1][j];
   /*17*/         printf("%d ", triangle[i][j]);
             }
   /*18*/     printf("\n");
         }
 }
```

给定链表，编写函数根据链表的长度动态分配一个数组，将链表元素存储于该数组。函数返回数组的首地址，并通过参数形式传回数组的元素个数。

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;

int* convertToArray(Node* head, int* size) {
     int length = 0;
     Node* curr = head; //链表头指针
     while (curr != NULL) { // while循环计算链表长度
         length++;
         curr = curr->next;
    }
     int* arr =  (1)
     (2)
      curr = head;
      int i = 0;
      while (curr != NULL) {
          arr[i] = curr->data;
          curr = curr->next;
        (3)
    }
     (4)
    }

int main() {
    Node *head = buildlist(); //构建整数链表，链表头指针为head。忽略其细节。
    int size; // 记录数组的元素个数
    int* arr = convertToArray(head, &size);
    return 0;
}
```

给定链表，编写函数根据链表的长度动态分配一个数组，将链表元素存储于该数组。函数返回数组的首地址，并通过参数形式传回数组的元素个数。

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* next;
} Node;

int* convertToArray(Node* head, int* size) {
     int length = 0;
     Node* curr = head; //链表头指针
     while (curr != NULL) { // while循环计算链表长度
         length++;
         curr = curr->next;
    }
     int* arr = （1）(int*)malloc(length * sizeof(int));
     (2) *size = length;
      curr = head;
      int i = 0;
      while (curr != NULL) {
          arr[i] = curr->data;
          curr = curr->next;
         (3) i++;
    }
     （4）return arr;
}

int main() {
    Node *head = buildlist(); //构建整数链表，链表头指针为head。忽略其细节。
    int size; // 记录数组的元素个数
    int* arr = convertToArray(head, &size);
    return 0;
}
```

2. 在一个未排序的整数数组nums中，找出其中没有出现的最小的正整数。如int nums[] = {3, 2, 1, 4, 1, –1}，则结果为5。结题思路是通过修改原数组来记录正整数的出现情况。首先将所有负数和零都标记为无效。然后遍历数组，将出现的正整数对应的数组位置（正整数i对应的数组位置是i–1）标记为相反数。注意对一个只有k个元素的数组，无需处理大于k的正整数。最后，在数组中找到第一个未被标记的索引位置。如果所有正整数（1到k）都出现了，返回数组长度加1。

函数调用中，nums为数组首地址，numsSize为数组元素个数。

```
#include <stdio.h>
#include "math.h"
int findMissingPositive(int* nums, int numsSize) {
    for (int i = 0; i < numsSize; i++) {  // for循环处理负数和0
        if (nums[i] <= 0) {
            nums[i] = numsSize + 1;
        }
    }
    for (int i = 0; i < numsSize; i++) {
        int num = abs(nums[i]); //abs为求绝对值函数
        if ( （5） ) {
            （6）        }
    }
    for (int i = 0; i < numsSize; i++) {
        if (nums[i] > 0) {
            （7）        }
    }
    （8）}
```

2. 在一个未排序的整数数组nums中，找出其中没有出现的最小的正整数。如int nums[] = {3, 2, 1, 4, 1, –1}，则结果为5。结题思路是通过修改原数组来记录正整数的出现情况。首先将所有负数和零都标记为无效。然后遍历数组，将出现的正整数对应的数组位置（正整数i对应的数组位置是i–1）标记为相反数。注意对一个只有k个元素的数组，无需处理大于k的正整数。最后，在数组中找到第一个未被标记的索引位置。如果所有正整数（1到k）都出现了，返回数组长度加1。

函数调用中，nums为数组首地址，numsSize为数组元素个数。

```
#include <stdio.h>
#include "math.h"
int findMissingPositive(int* nums, int numsSize) {
    for (int i = 0; i < numsSize; i++) {  // for循环处理负数和0
        if (nums[i] <= 0) {
            nums[i] = numsSize + 1;
        }
    }
    for (int i = 0; i < numsSize; i++) {
        int num = abs(nums[i]); //abs为求绝对值函数
        if (  （5）num <= numsSize) {
            （6）nums[num – 1] = –abs(nums[num – 1]);
// 标记为相反数
        }
    }
    for (int i = 0; i < numsSize; i++) {
        if (nums[i] > 0) {
            （7）return i + 1; //返回值
        }
    }
    （8）return numsSize + 1; //返回值
}
```

编写函数，用于将一个未排序的数组转成一个升序排列的链表。如给定数组int arr[] = {5, 2, 8, 1, 9}，则生成链表1->2->5->8->9。
请根据下面给出的部分代码编写函数arrayToLinkedList（函数中需要完成必要的如申请内存等工作）。

```
struct Node {
    int data;
    struct Node* next;
};
struct Node* arrayToLinkedList(int arr[], int size) {
    struct Node* head = NULL;
    ……
    ……
    return head;
}
int main() {
    int arr[] = {5, 2, 8, 1, 9};
    int size = sizeof(arr) / sizeof(arr[0]);
    struct Node* sortedList = arrayToLinkedList(arr, size);
    return 0;
}
```

根据如下的代码片段，编写函数replaceWord。将str中每次出现的子串oldWord替换成为新子串newWord。如下例，则输出结果为"Hello World! Hello Universe!"。注意新子串可能比原来的子串长或者短，因此可能需要移动后续的字符。为简化问题，假定str的长度总是足够的。

```
void replaceWord(char str[], char oldWord[], char newWord[]) ;
int main() {
    char str[1000] = "Hi World! Hi Universe!";
    char oldWord[] = "Hi";
    char newWord[] = "Hello";
    replaceWord(str, oldWord, newWord);
    printf("%s", str);
    return 0;
}
```

2. 下列表达式中值为0的是（ c ）。

A. 3%5　　B. 3/5.0　　C. 3/5　　D. 3.0/5

3. 在C 程序中，用（ b ）表示逻辑值“真”。

A. 1　　B. 非0的数　　C. 非 1 的数　　D. 大于 0 的数

8. 若a为int类型，且其值为3，则执行完表达式a-=a*a后，a的值是（ c ）

A. -3　B. 9　C. -6　D. 6

15. 设有以下说明语句

```
typedef struct
{ int n;
char ch[8];
}PER;
```

则下面叙述中正确的是（ b ）

A. PER 是结构体变量名　　B. PER 是结构体类型名

C. typedef struct 是结构体类型　　D. struct 是结构体类型名

17. 对于基类型相同的两个指针变量之间，不能进行的运算是（ c ）

A. <　　B. =　　C. +　　D. -

22. 设 x、y、z 和 k 都是 int 型变量,则执行表达式:x=(y=4,z=16,k=32)后,x 的值为 （ c ）

A. 4　　B. 16　　C. 32　　D. 52

27. 下列程序的输出结果是(c)。

```
main()
{ double d=3.2;  int x,y;
  x=1.2;  y=(x+3.8)/5.0;
  printf("%d \n", d*y);
  }
```

A. 3　　B. 3.2　　C. 0　　D. 3.07

3、 若定义int a[][2]={{1},{2},{3},4}; 则sizeof(a)的返回值为B( ) 。

A．48　　B．32　　C．16　　D．24

7、有如下程序

```
#include<stdio.h>
int main()
{
    int x = 2,y ;
    do
        switch (x) {
        case 1:
        y = 0;
        case 2:
        y = 1;
        case 3:
            x+=2; continue;
            case 4:
                x += y--;
            default:
                x++;
    }while(y>=1);
    return 0;
}
```

该程序执行完毕后，x的值为（ D ）。

A、3　B、4　C、5　D、6

以下程序的输出是？

```
#include <stdio.h>
int main() {
    int a = 10;
    int *p = &a;
    printf("%d\n", *p++);
    return 0;
}
```

A. 10
B. 地址错误
C. 编译报错
D. 不确定的值

A

以下程序的输出是？

```
#include <stdio.h>
int main()
{
printf("%5d\n", 123456);
return 0;
}
```

A. 12345
B. 123456
C. 23456
D. 0

B

下列结果的输出是

```
#include <stdio.h>
int i;
void prt()
{
    for (i = 5; i < 8; i++)
        printf("%c", '*');
    printf("\t");
}
int main()
{
    for (i = 5; i <= 8; i++)
        prt();
    return 0;
}
```

***

4.test.c文件中包括如下语句，文件中定义的四个变量中，是指针类型的变量为：【多选】

```
#define INT_PTR int*
typedef int* int_ptr;
INT_PTR a, b;
int_ptr c, d;
```

正确答案：acd

因为#define是宏定义，仅仅是直接替换,INT_PTR a, b; 进行宏替换后代码是这样的：int *a, b;这里的int *是a的类型，b的类型是int，故此次b只是int类型

6. 以下程序段输出结果是什么：

```c
#include<stdio.h>
int main()
{
    char s[] = "\\123456\123456\t";
    printf("%d\n", strlen(s));
    return 0;
}
```

正确答案：12

这里考查转义字符，注意：\\ 表示字符'\'，\123(ASCII码为83)表示字符'S'，\t表示制表符，这些都是一个字符

7. 下面代码段的输出是：

```
#include <stdio.h>
int main()
{
    int a=3;
    printf("%d\n",(a+=a-=a*a));
    return 0;
}
```

正确答案：-12

a+=a-=a*a等价于a=a+(a=a-a*a)，即先计算``a=a-a*a，所以此时a的值为3-3*3=-6，再计算-6+(-6)=-12赋值给a，所以a`的值 为-12，也就是整个表达式的值，就是-12

9. 执行下面的程序段，语句3的执行次数为（ ）

```
1 or(i = 0; i <= n-1; i++)    // (1)
2     for(j = n; j > i; j--)  // (2)
3         state;              // (3
```

正确答案：n(n+1)/2

外循环有n次，当i=0,内循环为n次,当i=1,内循环为n-1次,当i=2时，内循环为n-2次，以此类推，总次数为 n+(n-1)+(n-2)+…+2+1,就是个等差数列，等于n(n+1)/2

11.设变量已正确定义，以下不能统计出一行中输入字符个数（不包含回车符）的程序段是（ ）

A: n=0;while(ch=getchar()!='\n') n++;

B: n=0;while(getchar()!='\n') n++;

C: for(n=0;getchar()!='\n';n++);

D: n=0;for(ch=getchar();ch!='\n';n++);

正确答案：D
对于for循环，其中第一项初始化表达式只执行一次，因此ch只从输入流中取一个字符，之后就再不会取字符，因此会死循环

11.设变量已正确定义，以下不能统计出一行中输入字符个数（不包含回车符）的程序段是（ ）

A: n=0;while(ch=getchar()!='\n') n++;

B: n=0;while(getchar()!='\n') n++;

C: for(n=0;getchar()!='\n';n++);

D: n=0;for(ch=getchar();ch!='\n';n++);

正确答案：D
对于for循环，其中第一项初始化表达式只执行一次，因此ch只从输入流中取一个字符，之后就再不会取字符，因此会死循环

12. 若运行以下程序时，从键盘输入 `ADescriptor<回车>` ，则下面程序的运行结果是（）

```c
#include <stdio.h>
int main()
{
    char c;
    int v0=0,v1=0,v2=0;
        do
        {
            switch(c=getchar())
            {
                case'a':case'A':
                case'e':case'E':
                case'i':case'I':
                case'o':case'O':
                case'u':case'U':v1 += 1;
                default: v0+= 1;v2+=1;
            }
        }while(c!='\n');
    printf("v0=%d,v1=%d,v2=%d\n",v0,v1,v2);
    return 0;
}
```

正确答案：12 4 12

密码检查

小明同学最近开发了一个网站，在用户注册账户的时候，需要设置账户的密码，为了加强账户的安全性，小明对密码强度有一定要求：

密码只能由大写字母，小写字母，数字构成；
密码不能以数字开头；
密码中至少出现大写字母，小写字母和数字这三种字符类型中的两种；
密码长度至少为8

现在小明受到了n个密码，他想请你写程序判断这些密码中哪些是合适的，哪些是不合法的。

输入描述：

输入一个数n，接下来有n(n≤100)行，每行一个字符串，表示一个密码，输入保证字符串中只出现大写字母，小写字母和数字，字符串长度不超过100。

输出描述：

输入n行，如果密码合法，输出YES，不合法输出NO