# 答案

1- 回车（光标跳到当前行首）

水平制表（HT）

`\”`

`\\`

2- 没有，短路与

3- 7

4- static 修饰外部变量，改变了它的作用域（文件作用域）；函数内的变量，改变了它的生存期

5- 字符串数组

指向整型变量的指针数组

6-否

是

不能改变原左值数组，数组名指针指向了其它位置；而通过函数，是创建了一个新的指针临时变量。

7- s(1+4=5 字节) u(max(4,8) = 8 字节)

8- 有，返回了指向内部临时数组的指针

9- 可以

struct 默认权限是public, class 是 private

10- 多态

**第一道大题**

```
int a[100] = {-1};

int cmp(const void *a,const void *b)
{
    return(*(int *)b-*(int *)a);  //实现的是降序排序
}

long moreBigger(long in){
    int index = 0;
    do{
        a[index] = in %10;
        index++;
    }while(in /= 10);
```

```
14
15      qsort(a,100,sizeof(a[0]),cmp);  ///  <stdlib.h>
16      long res = 0;
17      for(int j =0; j < index; j++){
18          res = res*10 + a[j];
19      }
20
21      return res;
22  }
```

**第二道大题**

```
1   bool Judge(){
2       char str[100];
3       printf("Enter a message:");
4   //    gets(str);  ///  不能用 scanf
5       fgets(str, (sizeof str / sizeof str[0]), stdin);
6       char * p = strrchr(str, '\0');
7
8       printf("%d",p - str);
9       char newStr[100];
10      int p_newStr = 0;
11      for(int i=0; i < p - str; i++){
12          if( (str[i] >='a' && str[i] <= 'z' || str[i] >= 'A' && str[i] <=
    'Z' )){
13                  newStr[p_newStr] = str[i];
14                  p_newStr++;
15          }
16      }
17      newStr[p_newStr] = '\0';
18      printf("%s", newStr);
19
20      char * pp = strrchr(newStr, '\0');
21      char * end = pp-1;
22      char * begin = newStr;
23      bool  flag = true;
24      while (begin < end){
25          if( *begin != *end ){
26              flag = false;
27              break;
28          }else{
29              begin++;
30              end--;
31          }
32
33      }
34      return  flag;
35
```

```
}
```

**第三道大题**

```
void find(){
    int *array;
    int N;
    printf("请输入所要创建的一维动态数组的长度：");
    scanf("%d",&N);
    array=(int*)calloc(N+1,sizeof(int));

    for(int i=2; i <=N; i++){
        array[i] = 0;
    }

    for(int i=2; i<=N; i++){
        if(!array[i]){
            for(int j = i; i*j <=N; j++)
                array[i*j] = 1;
        }
    }

    int j=0;
    for(int i=2;i<=N;i++)
    {
        if(!array[i])
        {
            printf("%d ",i);
            j++;
            if(j%10 == 0)
                printf("\n");
        }
    }
    printf("\n");
    free(array);
}
```

**第四道题**

```
int main() {
    double precise;
    scanf("%lf", &precise);
    double res = 0;
    int index = 1;
    long long prod = 1;
    if(precise > 0){
        while(1){
```

```
            prod = prod * index;
            double current = 1.0 / prod;
            if( current < precise ){
                break;
            }
            res = res + current;
            index++;
        }
        res = res + 1;
        printf("%lf", res);
    }else{
        printf("Precision must be positive float");
    }

    return 0;
}
```

**第五道题**

```
typedef struct NODE
{
    char* val;
    struct NODE *next;
}node;//define node

node * head=NULL;//init head_node of stack
int length_stack(node *link);

//create node
node *create_node(char* val)
{
    node *p=(node *)malloc(sizeof(node));
    p->val=val;
    p->next=NULL;
    return p;
}

//push 头插法
node *push(char* val)
{
    node *p=create_node(val);
    p->next=head;
    head=p;
    return head;
}

//peek
char* peek(node *link)
```

```
{
    if(length_stack(link)==0)
    {
        printf("此栈为空");
        return 0;
    }
    return head->val;
}

//calculate length_length
int length_stack(node *link)
{
    int count=0;
    while(link)
    {
        count++;
        link=link->next;
    }
    return count;
}


//pop
char* pop(node *link)
{
    node *p;
    p=head;
    head=head->next;
    free(p);
    return p->val;
}
//printf_stack
void print_stack()
{
    while(head)
    {
        printf("%s   ",head->val);
        head=head->next;
    }
}
```

**第六道题**

```
void transform(double* source, double* dest, size_t n, double (*func)(double) ){
    for( int i=0; i< n; i++ ){
        dest[i] = func(source[i]);
    }
}
```

**第七道题**

```
typedef int datatype;
typedef struct node{
    datatype id;
    datatype grade;
    node* lchild = nullptr;
    node* rchild = nullptr;
    node* parent = nullptr;
    node(datatype id, datatype grade,node* lchild, node* rchild, node* parent):

 id(id),grade(grade),lchild(lchild),rchild(rchild),parent(parent){}

    node(datatype id, datatype grade): id(id),grade(grade){}
}Node;


void inorder(Node* in){
    if(in == nullptr)
        return;
    inorder(in->lchild);
    printf("id: %d \t grade: %d \n", in->id, in -> grade);
    inorder(in->rchild);
}
```

**第8道题**

```
class BaseGate{
public:
    BaseGate(){}
    BaseGate(double length, double width, double delay):length(length),width(width),delay(delay){}
protected:
    double length;
    double width;
    double delay;
public:
    virtual double getArea() =0;
    virtual double getDelay() = 0;
```

```
    virtual bool output() = 0;

};

class And: public BaseGate{
private:
    bool in1,in2;
public:
    Add(double length, double width, double delay, bool in1, bool in2):
            BaseGate(length,width,delay), in1(in1), in2(in2){}

    double  getArea(){
        return length * width;
    }

    double getDelay(){
        return delay;
    }

    bool output(){
        return in1 & in2;
    }

    /// 实现一个静态重载版本
    static bool  output(bool in1, bool in2){
        return in1 & in2;
    }
};

class Or: public BaseGate{
private:
    bool in1,in2;
public:
    Or(double length, double width, double delay, bool in1, bool in2):
            BaseGate(length,width,delay), in1(in1), in2(in2){}

    double  getArea(){
        return length * width;
    }

    double getDelay(){
        return delay;
    }

    bool output(){
        return in1 | in2;
    }

    /// 实现一个静态重载版本
```

```
    static bool  output(bool in1, bool in2){
        return in1 | in2;
    }
};


class Not: public BaseGate{
private:
    bool in1;
public:
    Not(double length, double width, double delay, bool in1, bool in2):
            BaseGate(length,width,delay), in1(in1){}

    double  getArea(){
        return length * width;
    }

    double getDelay(){
        return delay;
    }

    bool output(){
        return ~in1;
    }

    /// 实现一个静态重载版本
    static bool  output(bool in1){
        return ~in1;
    }
};


unsigned long simulate(BaseGate *gates[], unsigned int len){
    unsigned  long  res = 0;
    for(int i = 0; i < len ; i ++){
        gates[i] -> output();
        res += gates[i] -> getArea();
    }
    return res;

}
```