1、 任意进制转换问题（知识点：函数，进制转换）

【参考答案】

```
#include <iostream>
using namespace std;

void printInt(int n, int base);//函数声明
char baseChar[22]="0123456789ABCDEFGHIJ";
// 用此常量字符数组存储 n 进制对应的各个字符，可简化代码。
int main()
{
	int x;
	cout << "Please Input a number:" << endl;
	cin >> x;
	printInt(x, 2);
	cout << endl;
	printInt(x, 8);
	cout << endl;
	printInt(x, 16);

	return 0;
}

/*
*		printInt 用于打印整数 n 的 base 进制表示
*/

void printInt(int n, int base)
{
	int highn, numOfDigits=0;
	char result[100];
	//用于逆序存储转换后的结果，个位在前。十位在后，。。。
	if(n<0) {
		cout << "-"; n=-n;
	}
	while(n>=base){
		result[numOfDigits]=baseChar[n%base];		// 注意存储的是字符
		numOfDigits++; n=n/base;
	}

	result[numOfDigits]=baseChar[n];		// 注意存储的是字符
	numOfDigits++;

	for(int i=numOfDigits-1;i>=0;i--)
		cout << result[i];
}
```

注：还可以用递归来实现函数 printInt，其递归版本更加简洁，代码如下：

```
void printInt(int n, int base)
{
	if(n<0) {
		cout << "-";		// 打印符号
		n=-n;	// 转换为正数
```

```
    }
    if(n<base) //此时 n 只有个位数，直接打印该 base 下的对应字符。
        cout << baseChar[n];

    else
    {
        printInt(n/base,base);          //递归打印 n 的十位以上的高位部分
        cout << baseChar[n%base]; //打印 n 的个位数字字符
    }
}
```

2、改写字符串（知识点：字符串，数组作为函数参数）

【参考答案】

```
#include<iostream>
#include<cstring>
using namespace std;
void replaceAll(char [],char,char);    // 注意函数声明中的参数名称可以不写

int main()
{
    char s1[81]="nannies"; char c1,c2;

    replaceAll(s1, 'd','n'); cout << s1;

    cout << "请输入字符串:"; cin.getline(s1, 80, '\n');

    cout << "请输入待替换的字符:"; cin >>c2;
    cout << "请输入替换后的字符:"; cin >>c1;

    replaceAll(s1, c1,c2); cout << s1;

    return 0;
}

void replaceAll(char str[],char c1,char c2)
{
    int n;
    n=strlen(str);

    for(int i=0; i<n; i++){  // 依次遍历字符串的每个字符
        if(str[i]==c2)
            str[i]=c1;     // 用 c1 替换 c2
    }
}
```

3、计算单词次数（知识点：字符串处理，数组，函数）

【参考答案】

```
#include<iostream>
#include<cstring>
```

```
#include <cctype>
#include <iomanip>

using namespace std;
int readWords();
void countAndPintWords(int numOfWords);
char words[1001][21];

int main()
{
    int totalNumOfWords;

    totalNumOfWords=readWords();
    countAndPrintWords(totalNumOfWords);
    return 0;
}

/* 函数 readWords() 从标准输入读入若干行文本，识别其中的单词，忽略标点符号 ，把识别得到的每个单词依次存入全局数组 words 中，函数返回输入文本中的单词的总数
*/

int readWords(){
    char str[81],word[21];
    int numOfWords=0;    //记录单词总数

    cout << "请输入字符串:";
    while(cin.getline(str, 80, '\n')){
    //读入一行字符串到 str 中，回车换行结束一行输入。
/* 如何结束循环：在一行输入时，直接按下 ctrl+Z 键，然后按回车，此时 cin.getline()的值是 0，可结束输入。（1）这里展示了我们如何利用 cin 读入未知数目的输入。关键在于利用失败的 cin.getline()返回 0，一种触发失败的方式就是主动输入文件结束符，而 Windows 系统下输入文件结束符的方式，是按 Ctrl + Z。在使用 cin>>输入数据时方法相同。（2）注意：如果你不用这种方法，也可以自己规定一个本来不会出现的特殊的标记字符，譬如'#', 来表示输入结束，结束循环。
*/
        int nc=strlen(str); //计算 str 的大小
        for(int i=0;i<nc;i++){
            //从左到右依次遍历 str 的每个字符，当前字符的下标为 i
            if(!isspace(str[i])&& !ispunct(str[i])){
            //若当前字符不是空白字符，也非标点符号
                int j=0;
            // j 指示正在访问的字符在当前单词中的位置
                while(!isspace(str[i])&&!ispunct(str[i])){
                    //若不是空白字符，也非标点符号
                    //把当前词逐个字符读入 word 中,按照小写字符写入
                    word[j]=tolower(str[i]);
                    i++;          //str 中的当前位置更新
                    j++;          //word 中的字符位置更新
                }
                word[j]='\0';      // 在 word 的当前位置设置字符串结束标志
```

```
                strcpy(words[numOfWords],word);
                // 把新得到的单词 word 拷贝到字符串数组中

                numOfWords++; //单词个数更新
            }
        }
    }
    return numOfWords;   //得到的单词总数
}

/*
*       函数 countAndPrintWords(int numOfWords) 从全局数组 words 中，
*       依次计算每个单词在该数组中出现的次数，
*       按照表格形式输出单词和对应的次数，每输出五个单词换行。
* 注：函数的输入参数 numOfWords 是数组 words 中存储有效单词的个数。
*/

void countAndPrintWords(int numOfWords){
    int Wcount;
    char word[21];
    int showPos=0;
    //用于控制打印显示格式，表示当前行已经打印的单词个数
    //从前到后依次考虑每个单词，当前为第 i 个单词
    for(int i=0;i<numOfWords;i++){
        if(!strcmp(words[i],"")) continue;
        //如果当前位置的单词为空字符串，忽略之
        strcpy(word,words[i]); // 将当前位置的单词 word[i]拷贝给 word
        // 以下首先计算与当前单词 word 相同的单词的个数
        Wcount=1;//记录与当前单词 word 相同的单词的个数，初始为 1
        for(int j=i+1;j<numOfWords;j++){
      //从当前位置 i 起，依次遍历,到最后一个单词
            if(!strcmp(word,words[j]))
              //如果遇到的单词 words[j]与 word 相同
            {
                Wcount++;       //
                strcpy(words[j],"");        //将其置为空串
            }
        }

        // 以下输出当前单词 words，以及相同单词的个数
        cout << word << setw(10-strlen(word)) << Wcount << "\t";
        // 以上 setw 用于控制紧随其后的输出所占的间隔

        showPos++;      // 当前行打印的单词个数+1
        if(showPos == 5) {          //控制每输出 5 个单词换行
            cout << endl; //换新行
            showPos = 0;//重置当前行的单词个数为 0
        }
    }
}
```

【注意】：
中间一段代码也可不用定义 char word[21]这个字符串，直接用原来的 words[i]即可，可修改代码如下：

```
//strcpy(word,words[i]);      这一句不再需要

// 计算与当前单词 word[i]相同的单词的个数
Wcount=1;          //记录与当前单词 word[i]相同的单词的个数，初始为 1
for(int j=i+1;j<numOfWords;j++){
 //从当前位置 i 起，依次遍历,到最后一个单词
      if(!strcmp(words[i],words[j]))
      //如果遇到的单词 words[j]与 word[i]相同
      {
            Wcount++;          //
            strcpy(words[j],"");          //将其置为空串
      }
}

// 以下输出当前单词 words[i]，以及相同单词的个数
cout << words[i] << setw(10-strlen(words[i])) << Wcount << "\t";
```

4、函数模板

【参考答案】

```
#include<iostream>
using namespace std;

template< typename T>
void selectSort(T arr[],int n);

template<typename T>
void selectSort2(T arr[],int n);

template<typename T>
void printArray(T arr[],int n);

int main()
{
      int intarr[10]={20,18, 7, 19, 9, 8, 2, 12,10,1};
      float floatarr[12]={9.1, 1.9, 8.2, 7.3, 6.4, 3.7, 5.5, 4.6, 2.8, 5.8, 4.6, 2.9};
      char chararr[8]= {'C', 'd', 'e','x','B','D','A', 'a' };

      selectSort(intarr,10);
      cout << "the sorted int array:"; printArray(intarr,10);

      selectSort(floatarr,12);
      cout << "the sorted float array:"; printArray(floatarr,12);

      selectSort(chararr,8);
      cout << "the sorted character array:"; printArray(chararr,8);

      return 0;
}
```

```
template<typename T>
void selectSort(T arr[],int n)
{
    for(int i=0;i<n-1;i++)
    {//外圈，比较次数 for(int j=i+1;j<n;j++)
        {       //每轮比较
            if(arr[i]>arr[j])
            {
                //将位置为 j 的元素与位置 i 的元素交换
                T temp=arr[i]; arr[i]=arr[j]; arr[j]=temp;
            }
        }
    }
}

//解法二：以上解法还可优化，每轮比较过程中，只需要记下这一轮最小元素// 的位置，最后交换一次即可
template< typename T>
void selectSort2(T arr[],int n)
{
    for(int i=0;i<n-1;i++)
    {//外圈，比较次数
        int min=i; // min 记录本轮比较最小元素的位置
        for(int j=i+1;j<n;j++) //每轮比较 if(arr[min]>arr[j])
            min=j;      //记录当前最小元素的位置

 //交换位置为 min 的最小元素与位置 i 的元素，即本轮里最左边的元素）
        T temp=arr[i]; arr[i]=arr[min]; arr[min]=temp;
    }
}

template< typename T>
void printArray(T arr[],int n)
{
    for(int i=0;i<n-1;i++)
        cout << arr[i] << ", ";      //两个元素之间用逗号隔开
    cout << arr[n-1] << endl;
    //为避免最后元素的后面出现多余的分隔符，单独输出最后一个元素
}
```

【注】以上的 typename 也可以换成 class

5、回文判定（知识点：递归求解问题）

【参考答案】

```
#include <iostream>
#include <cctype>
#include <cstring>
using namespace std;

bool isPalindrome(char str[]);
bool isPalindromeRec(char str[],int start,int ending) ;
```

```
int main()
{
	char str[81];
	cout << "please input a string( < 80 characters):";
	cin.getline(str,80,'\n');

	if(isPalindrome(str))
		cout << str << " is a palinedrome.";
	else
		cout << str << " is not a palinedrome.";
	return 0;
}

bool isPalindrome(char str[])
// 封装函数：该函数将递归的函数 isPalindromeRec 封装起来
{
	return isPalindromeRec(str,0,strlen(str)-1);
	// 提供递归所需要的参数：指示字串的下标范围。

}

bool isPalindromeRec(char str[],int start,int end)
// start,end 为元素的下标，分别为子串中的第一个元素和最后一个元素。
{
	int n=end-start+1;	//计算当前所考虑子串的长度

	if((n==0) ||(n==1)) return true;

	if(toupper(str[start])!=toupper(str[end]))
		// 用 touper()函数将字符转换为大写，然后再比较。
		// 注意：toupper(str[start]) 这个表达式是大写字符，但是
		// str[start]本身并未做转换。
		return false;
		return	isPalindromeRec(str,start+1,end-1);
		// 递归：转换为对子串判断
	}
```

6、计算数字根（知识点：递归求解问题）

【参考答案】

```
#include<iostream>
using namespace std;
int digitroot(int ); // 函数声明

int main()
{
	int n;
	do{
		cout << "please input an integer:" << endl; cin >> n;
	} while(n<=0);
```

```
        cout << digitroot(n); return 0;
}

int digitroot(int n)
{
        int n1;
        if(n<10) return n;          // Base case, 结束递归的条件
        n1 = digitroot(n/10)+ n%10;
        // 转化为小规模问题，递归,注意 n/10 < n
        return digitroot(n1);
        // 转化为小规模问题，注意 n1 < n
}

// 注：以上代码还可进一步简化为
int digitroot(int n)
{
        if(n<10) return n;
        return digitroot(digitroot(n/10)+ n%10);
}
```

7. 八后问题 Eight Queens Puzzle（知识点：递归求解问题）

```
#include<iostream>
using namespace std;
int const MaxRowNum=8;    //   最大行数
int const MaxColNum=8;     //   最大列数
int col[20],count=0;

void printLayout(){ cout<<"摆放布局
    "<<count<<":"<<endl; for( int i=0;
    i<MaxRowNum; i++) {
       for( int j=0; j<MaxColNum;
          j++) { if( j == col[i] )
          cout<<"1 ";
          else cout <<"0 ";
```

```
        }
        cout<<endl;
    }
    cout<<endl;
}

void searchLayout(int
row){
    if (row ==
        MaxRowNum){ pr
        intLayout();
        ++count;
        return;
    }
    for(int i=0; i<MaxColNum;
        ++i){ col[row] = i;
        bool done = true;  //   表示当前行是否摆放成功
        // 检查当前行摆放queen 的位置(row,col[row])与先前的queens 是否冲突
        for (int preRow=0; preRow<row; ++preRow)
          if (col[row] == col[preRow]||row-preRow ==col[row]-
              col[preRow]|| row-preRow == col[preRow]-
              col[row]) {
            done = false;
            break;
          }
        if (done) searchLayout(row+1); //   第row 行摆放成功，开始摆放第row+1
        行
    }
}
int main(){
    searchLayout(0);
    cout<<count<<en
    dl; return 0;
}
```