1 编写一个判断素数的函数，主函数中利用这个函数，打印出n与m之间的所有素数。

#include <iostream>

using namespace std;

int prime(int x)

{

int i,flag=1;

for(i=2;i<x;i++)

if(x%i==0)

flag=0;

return flag;

}

int main()

{

int m,n,i,count;

while(cin>>n>>m)

{

count=0;

for(i=n;i<=m;i++)

if(prime(i))

if(count==0)

{

cout<<i;

count++;

}

else

cout<<" "<<i;

cout<<endl;

}

return 0;

}

2 输出所有的三位水仙花数，判断是否是一个“水仙花数”，水仙花数是指3位数的各位数字的立方和等于这个三位数本身。例如153=1\*1\*1+5\*5\*5+3\*3\*3。

#include <iostream>

using namespace std;

int main()

{

int i;

int b, c, d;

i = 100;

do

{

b = i % 10;//个位数

c = (i/10)%10;//十位数

d = i / 100;//百位数

if (b\*b\*b + c\*c\*c + d\*d\*d == i)

{

cout << i << endl;

}

i++;

} while (i<1000);

system("pause");

return 0;

}

3 给你一个包含 n 个整数的数组 nums，判断 nums 中是否存在三个元素 a，b，c ，使得 a + b + c = 0 ？请你找出所有和为 0 且不重复的三元组。

注意：答案中不可以包含重复的三元组。

例如

**输入：**nums = [-1,0,1,2,-1,-4]

**输出：**[[-1,-1,2],[-1,0,1]]

class Solution {

public:

    vector<vector<int>> threesums(vector<int>& nums)

{

vector<vector<int>> result;

int length = nums.size();

if (length < 3)//元素小于三个直接返回

return result;

sort(nums.begin(), nums.end());//按升序排列nums数组

for (int i = 0; i < length - 2; i++)//固定位i要留出begin和end的位置，总次数length-2

{

int left = i + 1;

int right = length - 1;

if (nums[i] > 0)

{

break;

}

if (i > 0 && nums[i] == nums[i - 1])//从第二次循环开始，如果固定位nums[i]和上一次相同，则跳过此次循环，目的是去重

{

continue;

}

while (left < right)

{

int sum = nums[i] + nums[left] + nums[right];

if (sum == 0)

{

result.push\_back({ nums[i],nums[left],nums[right] });

while (left < right && nums[left] == nums[left + 1])//去重

left++;

while (left < right && nums[right] == nums[right - 1])//去重

right--;

}

else if (sum < 0)

left++;

else

right--;

}

}

return result;

}

};

4 将两个升序链表合并为一个新的 升序 链表并返回。新链表是通过拼接给定的两个链表的所有节点组成的。

例如

输入：l1 = [1,2,4], l2 = [1,3,4]

输出：[1,1,2,3,4,4]

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode():val(0),next(nullptr){}

\* ListNode(int x) : val(x), next(nullptr) {}

\* ListNode(int x,ListNode\*next):val(x),next(next){}

\* };

\*/

class Solution {

public:

ListNode\* mergeTwoLists(ListNode\* list1, ListNode\* list2) {

ListNode \*newHead = new ListNode(-1);

ListNode\* tmp = newHead;

while (list1 && list2)

{

if (list1->val < list2->val)

{

tmp->next = list1;

tmp = tmp->next;

list1 = list1->next;

}

else

{

tmp->next = list2;

tmp = tmp->next;

list2 = list2->next;

}

}

//未完数据

tmp->next = list1 ? list1 : list2;

return newHead->next;

}

};

5 实现 strStr() 函数。

给你两个字符串 haystack 和 needle ，请你在 haystack 字符串中找出 needle 字符串出现的第一个位置（下标从 0 开始）。如果不存在，则返回  -1

例如：

输入：haystack = "hello", needle = "ll"

输出：2

class Solution {

public:

    int strStr(string haystack, string needle) {

int NL = strlen(needle);

int HL = strlen(haystack);

if(NL<1)

return 0;

int countN,countH;

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

{

if(HL-i<NL)

return -1;

countH =i;

countN =0;

while( haystack[countH++] == needle[countN++])

if(countN == NL)

return i;

}

return -1;

}

    }

};

6 给你一个有序数组nums，请你原地删除重复出现的元素，使每个元素只出现一次，返回删除后数组的新长度。不要使用额外的数组空间，必须在原地修改输入数组并在使用 O(1) 额外空间的条件下完成。

class Solution {

public:

int removeDuplicates(vector<int>& nums) {

int k=0;

for(int i=0;i<nums.size();i++){

if(!i||nums[i]!=nums[i-1]){

nums[k++]=nums[i];

}

}

return k;

}

};

7给你一个链表，删除链表的倒数第 n 个结点，并且返回链表的头结点。

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode() : val(0), next(nullptr) {}

\* ListNode(int x) : val(x), next(nullptr) {}

\* ListNode(int x, ListNode \*next) : val(x), next(next) {}

\* };

\*/

class Solution {

public:

ListNode\* removeNthFromEnd(ListNode\* head, int n) {

ListNode result = head;

ArrayList<ListNode> list = new ArrayList<>();

while (head.next != null) {

list.add(head);

head = head.next;

}

list.add(head);

if (list.size() == n) return result.next;

ListNode before = list.get(list.size() - n - 1);

if (n == 1) {

before.next = null;

return result;

}

//ListNode remove = list.get(list.size()-n);

ListNode after = list.get(list.size() - n + 1);

before.next = after;

return result;

}

};

8用同一个函数名对10个数据进行从小到大进行排序，数据类型可以是整形、单精度、双精度。用重载函数和模板各实现一遍，要求使用快速排序法，在屏幕上输出实时排序的情况，要求使用控制台模式，用字符个数代表数据大小。

//函数模板实现

#include <iostream>

using namespace std;

#define n 5

template <typename T>

void sort(T a[])

{

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

for(int j=i+1;j<n;j++)

if(a[i]> a[j]){

T t=a[i];

a[i]=a[j];

a[j]=t;

}

}

int main()

{

int i,a1[5]={1,3,2,6,5};float a2[5]={1.2,1.1,2.2,3.1,2.6};double a3[5]={1.8,1.7,1.9,2.1,2.0};

sort(a1);

sort(a2);

sort(a3);

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

cout <<a2[i]<< " ";

cout <<endl;

system("PAUSE");

return 0;

}

//重载函数实现

#include <iostream>

using namespace std;

float queue(float\*a,int n)

{

float t;

int i,j;

for(i=1;i<n;i++)

{

for(j=0;j<n-i;j++)

{

if(a[j]>a[j+1])

{

t=a[j+1];

a[j+1]=a[j];

a[j]=t;

}

}

}

for(i=0;i<n;i++)

{

cout<<a[i]<<" ";

}

cout<<endl;

return 0;

}

int queue(int\*a,int n)

{

int t;

int i,j;

for(i=1;i<n;i++)

{

for(j=0;j<n-i;j++)

{

if(a[j]>a[j+1])

{

t=a[j+1];

a[j+1]=a[j];

a[j]=t;

}

}

}

for(i=0;i<n;i++)

{

cout<<a[i]<<" ";

}

cout<<endl;

return 0;

}

double queue(double\*a,int n)

{

double t;

int i,j;

for(i=1;i<n;i++)

{

for(j=0;j<n-i;j++)

{

if(a[j]>a[j+1])

{

t=a[j+1];

a[j+1]=a[j];

a[j]=t;

}

}

}

for(i=0;i<n;i++)

{

cout<<a[i]<<" ";

}

cout<<endl;

return 0;

}

int main()

{

int a[5]={22,13,26,55,5};

float b[5]={1.2f,3.4f,2.2f,5.4f,4.5f};

double c[5]={1.2201,3.2121,0.3322,5.4433,2.1122};

queue (a,5);

queue (b,5);

queue (c,5);

}

9 划拳游戏，15个男生和15 个女生围成一圆圈，从第一个人开始依次报数，每数到第9个人就喝酒，如此循环进行直到仅余15个人为止。用程序求出女生怎么排位，才能使每次喝酒的都是男生。

#include<iostream>

#include<stdlib.h>

using namespace std;

void make(int \*base,int n,int pos,int c,int m)

//参数的意义,base数组名，n数组长度。pos跑格的一个东西。c计算次数的。m每次跑路的长度。

{

int j=0;

cout<<"NO. "<<++c<<" 第"<<pos+1<<"位出列"<<endl;//输出

base[pos]=0;//踢掉

if(c==n)return; //出口

while(j-m)if(base[pos=(pos+1)%n])j++;//递归点 ,每次数到几这个3就改到几

make(base,n,pos,c,m);//递归

}

int main()

{

int n,m,c=0,pos;//从N开始数,则把pos改为N-1就行了.

cout<<"请输入总人数"<<endl;

cin>>n;

cout<<"请输入要隔几个人:"<<endl;

cin>>m;

int \*base=new int[n];

for(int i=0;i<n;i++)base[i]=1;

pos=m-1;

make(base,n,pos,c,m);

delete[]base;

system("PAUSE");

return 0;

}

10 完成一个时间类，有成员年、月、日、小时、分钟、秒，实现三种构造函数，一种根据时间并带有默认时间值，一种根据日期+秒速，一种拷贝构造函数，有两个显示时间的成员函数（24小时制和12小时制），完成主函数，生成两个时间的对象，并设置不同的时间，输出显示时间，并计算时间之间的差，使用QT或MFC，在桌面上显示一个带有表针的表面。

#include<iostream>

using namespace std;

class clock {

private:

int hour, minute, second;

public:

clock(int a, int b, int c) {

hour = a; minute = b; second = c;

}

~clock() {

cout << "释放内存" << endl;

}

void display();

void settime(int a, int b, int c) {

hour = a; minute = b; second = c;

}

clock(clock& p) {

hour = p.hour; minute = p.minute; second = p.second;

}

}

void clock::display() {

cout << "时间为：" << hour << "时" << minute << "分" << second << "秒" << endl;

}

int main() {

clock a(23, 3, 1);

a.display();

a.settime(17, 05, 52);

clock b(a);

b.display();

}

11 使用QT实现橡皮筋功能，其中起始点在坐标（50，50），鼠标移动，起始坐标点到鼠标点的连线，鼠标左键，起始点换成当前鼠标点，连线使用红色，线粗1.5倍，同时在界面的右下角实时刷新显示鼠标坐标；并实现菜单栏，在菜单栏上有两个菜单项，一个是距离，点击以后跳出一个对话框，输出线段的距离和，一个是退出，点击以后程序退出。要求程序每行有注释。

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QMenuBar>

#include <QDialog>

#include <QMessageBox>

#include <QMouseEvent>

#include <QWidget>

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(*parent*)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);//对话框

connect(ui->actionjuli,&QAction::triggered,[=](){

*//QDialog* *dlg(this);*

*//dlg.resize(200,100);*

*//dlg.exec();*

QMessageBox::warning(*this*,"warning","距离");

});

connect(ui->actiontuichu,&QAction::triggered,*this*,&QWidget::close);//退出

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

int **main**(int argc, char \*argv[])

{

QApplication a(*argc*, *argv*);

QTranslator translator;

*const* QStringList uiLanguages = QLocale::system().uiLanguages();

*for* (*const* QString &locale : uiLanguages) {

*const* QString baseName = "Qt3\_" + QLocale(locale).name();

*if* (translator.load(":/i18n/" + baseName)) {

a.installTranslator(*&translator*);

*break*;

}

}

MainWindow w;

w.show();

*return* a.exec();

}