上 海 交 通 大 学 试 卷

（ 2017 至 2018 学年 第2学期 ）

班级号\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 学号\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 姓名

课程名称 C程序与算法设计 成绩

说明：程序除非注明使用C++的情况外，请用标准C语言描述，注意节省卷面答题空间。

一、（20分）填空（答案直接填在空格处）

1. 给出C语言中下列转义字符的意义，"\r":回车（光标跳到当前行首） ；"\t": 水平制表（HT）；给出C语言中双引号的字符串表示： \” ；反斜杠的字符串表示： \\
2. 对于一个整数i和j，语句if(i&&(j/i>0)) j++;的写法有无出现除0异常的潜在问题？为什么？ 没有，短路与
3. 已知int i=5, j=4, k=7; 表达式i---j--+--k;的值是多少： 7
4. 在C中，以static修饰外部变量与函数内部变量的作用分别是什么：

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

1. 声明char\* a, b[10], c;中b的数据类型是什么？ 字符串数组

声明int \*day[7]中day的具体类型是什么： 指向整型变量的指针数组

1. 代码片段{int a[]={1,2}; printf("%d",\*a++);}是否正确： 否 ；假设有int a[]={1,2};函数定义int fun(int a[]){return \*a++;}的写法是否正确： 是

得出上述结论的理由是什么？ 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。

1. 已知struct s\_tag{char a; int b;} s; union u\_tag{int a; double b;} u;试比较在32位的机器上，s与u所占据的内存空间大小并说明原因：

s(1+4=5 字节) u(max(4,8) = 8 字节)

1. 函数定义 int\* fun(){int a[]={1,2}; int \*b=a; return b++;}有无问题？如有问题请列出具体的问题： 有，返回了指向内部临时数组的指针
2. 在C++中，struct关键字是否可以用于声明一个类？ 可以 ；与用class关键字声明一个类的差别是什么？ struct 默认权限是public, class 是 private
3. 在C++中，虚函数的主要用途是什么？ 多态

； 纯虚函数和虚函数的主要区别是什么？

纯虚函数不要求实现，且不可以直接声明该类的对象，但可由子类来实例化； 有纯虚函数的类是抽象类；

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 |  |  |  |  |  |  |  |  |  |  |
| 得分 |  |  |  |  |  |  |  |  |  |  |
| 批阅人(流水阅  卷教师签名处) |  |  |  |  |  |  |  |  |  |  |

**我承诺，我将严格遵守考试纪律。**

**承诺人：**

二、（80分）综合题

1. （10分）用C编写一个函数，该函数的参数为一个长整数，返回值亦为长整数。该函数要求的功能为：将输入参数的各个数位拆出，然后重新排列组合为一个更大的长整数并返回该长整数。例如输入参数2314，拆出2、3、1、4，然后重新组合为最大的4321返回。编写该函数时无需考虑是否超出长整数的表示范围；只需考虑十进制形式。

|  |
| --- |
| 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);  qsort(a,100,sizeof(a[0]),cmp); /// <stdlib.h>  long res = 0;  for(int j =0; j < index; j++){  res = res\*10 + a[j];  }  return res;  } |

1. （10分）用C编写一个函数，用于进行回文(Palindrome，即从左往右和从右往左看是相同的句子)测试。该函数的输入为一条消息，输出是表示该消息是否为回文的标志。为简单起见，只考虑英文消息，而且忽略掉消息中所有不是字母的字符。运行时场景如下：

Enter a message: He lived as a devil, eh?

Palindrome

Enter a message: Madam, I am Adam

Not a palindrome

答案是不区分大小写的

|  |
| --- |
| bool Judge(){  char str[100];  printf("Enter a message:");  // gets(str); /// 不能用 scanf  fgets(str, (sizeof str / sizeof str[0]), stdin);  char \* p = strrchr(str, '\0');  printf("%d",p - str);  char newStr[100];  int p\_newStr = 0;  for(int i=0; i < p - str; i++){  if( (str[i] >='a' && str[i] <= 'z' || str[i] >= 'A' && str[i] <= 'Z' )){  newStr[p\_newStr] = str[i];  p\_newStr++;  }  }  newStr[p\_newStr] = '\0';  printf("%s", newStr);  char \* pp = strrchr(newStr, '\0');  char \* end = pp-1;  char \* begin = newStr;  bool flag = true;  while (begin < end){  if( tolower(\*begin) != tolower(\*end )){ /// include <ctype.h>  flag = false;  break;  }else{  begin++;  end--;  }  }  return flag;  } |

1. （10分）有一种寻找素数的算法称为埃拉托斯特尼筛算法，下面给出了这一算法的具体流程。请用C语言编写一个函数，实现埃拉托斯特尼筛算法，并将找到的所有素数返回。埃拉托斯特尼筛算法需要提供一个正整数参数N，则该算法将找到所有小于N的素数。算法具体流程如下：
2. 定义整数数组P，P的大小为N+1，将下标为范围内的P[i]设置为0；
3. 设定i的初值为2；
4. 若，则算法终止；
5. 若P[i]为0，则为素数；
6. 对于所有满足的的所有正整数值，设定P[]为1；
7. 令加1，转步骤3。

|  |
| --- |
| void find(){  int \*array;  int N;  printf("请输入正整数N：");  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);  } |

1. （8分）数学常量的值在数学上可以用一个无穷级数表示：

请编写一个C函数，用下面公式计算的近似值：

该函数需要用户输入一个参数，当小于时，上述级数的计算终止，并返回计算得到的近似值。 编写代码时注意避免重复计算。

|  |
| --- |
| 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;  } |

1. （10分）堆栈(Stack)是一种很常用的数据结构，它是一种后进先出的数据结构，支持两种基本操作：压栈(Push)和出栈(Pop)。其中，Push将数据放入堆栈的顶端，栈顶top指针加一；Pop操作将顶端数据弹出，栈顶top指针减一。课上的Stack设计采用了C数组实现，但C数组的大小不能变化，导致以数组设计的Stack不够灵活。请在此处给出基于链表实现的Stack数据结构。代码用C实现（可能要包括链表节点、Stack、栈顶、栈底、Push操作、Pop操作等，同时需灵活运用结构等知识）。假定需要在Stack中存储英文姓名(字符串)，即设计一个存放字符串的栈，Push操作将一个姓名字符串压入栈中，Pop操作将最后压入栈的字符串返回。



|  |
| --- |
| 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;  }  } |

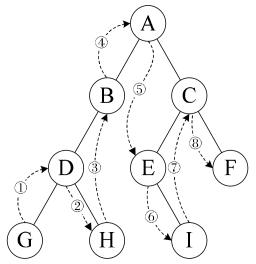
1. （8分）用C编写一个名为transform()的函数，接受4个参数，包括一个内含double类型数据的源数组名source、一个内含double类型数据的目标数组名dest、一个表示数组元素个数的int类型参数(假设前两个数组尺寸相同)、一个函数(指针)。transform()函数把指定的第四个参数(函数指针)应用于源数组中的每个元素，并把所应用函数的返回值保存于目标数组同样下标的位置。例如调用**transform(source, dest, 100, sqrt);**将source数组中的每个元素计算平方根并保存于dest中。如果将sqrt换做sin，则计算source中每个元素的正弦函数值并储存到dest中。请给出transform()函数的代码。假设其第四个参数为一个接受一个double型输入参数，返回一个double型值的函数指针。

|  |
| --- |
| void transform(double\* source, double\* dest, size\_t n, double (\*func)(double) ){  for( int i=0; i< n; i++ ){  dest[i] = func(source[i]);  }  } |

1. （12分）在计算机科学领域，二叉树是一种具有多种优良特性的tree数据结构。Wiki上对其定义为：In computer science, a binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child. 现假设将某门课程所有学生的成绩放置在一棵二叉树中，如下图所示，即每个节点至少包含学号ID和成绩Grade两个属性，同时必须有左右两个孩子指针指向其左右孩子，可能还需要一个父指针指向父亲节点。同时，根节点root没有父节点，叶子结点没有子节点，节点也可能只有一个孩子。要求：1）请借鉴链表的定义来用C语言定义该二叉树数据结构；2）利用1)中的数据结构，假设已经有了一个二叉树，且树内部各节点都存储了有效数据，试编写函数按照中序遍历的次序遍历该树并以此打印每个学生的学号和成绩。



注：下图给出了二叉树中序遍历的过程，图中的编号即遍历的次序，即递归进行下述过程：

若二叉树为空，则算法结束；否则：

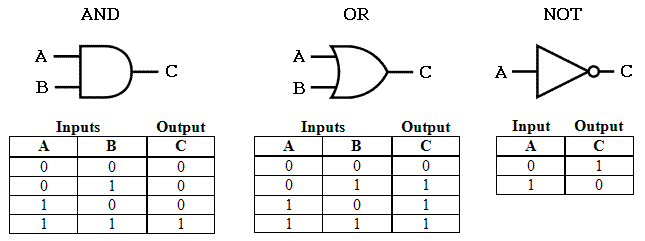
中序遍历根结点的左子树；

访问根结点；

中序遍历根结点的右子树。

|  |
| --- |
| 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);  } |

1. (12分) 假设我们正在用C++实现一套EDA软件，该EDA软件可以实现基本的组合逻辑门电路的功能仿真。简单起见，组合逻辑门电路中只包含3类门电路，分别是与门、或门、非门。每个门的输入、输出引脚代表1 bit信息。与门的输出是所有输入的"and"运算的结果；或门的输出是其所有输入的"or"运算的结果；非门的输出是其输入的求反操作的结果，具体请见下图。从集成电路设计的角度考虑，我们需要每个门至少记录其本身的长(纳米，nm)、宽(nm)、固有延迟(皮秒，ps)这三个属性，同时每个门需要提供返回面积、返回延迟、计算输出值这三个成员函数。与门只考虑2输入与门、或门也只考虑2输入或门、非门只有1输入，三类门的输出都是1 bit。请运用所学的C++面向对象编程的知识：
2. 将与/或/非等门设计为C++的Class，并形成类层次结构。请给出各个的类的完整代码。注意此处不可以将各类门设计成为毫无关系的独立Class，需要利用基类/派生类等关系进行设计。



|  |
| --- |
| 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;  }  }; |

(b) 设计一个函数simulate，该函数有两个参数，第一个参数是一个数组，第二个参数表示数组包含的元素个数。数组包含的元素为指向各门对象的指针。利用C++的多态机制，在此函数中依次计算各个门对象的输出值，并汇总数组包含的所有门对象的面积。假设各个门对象的输入值都已经准备好。simulate函数的可能原型为：

unsigned long simulate(Gate \*gates[], unsigned int len);

|  |
| --- |
| 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;  } |