一、读程题

1. int a,b,d=10;

   a = d | 1 % 9, b = d & (-1);

   printf("%d,%d",a,b);

   答案：11,10

   1%9=1

   a=10|1=(1010)|(0001)=(1011)=11

   b=10&(-1)=(1010)&(1111)=(1010)

2. int main(){

   int i;

   void f();

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

   f();

   return 0;

   }

   void f(){

   static int cnt = 0;

   ++cnt;

   Printf("%d",cnt);

   }

   答案：12345

   static int cnt是静态局部变量

3. int a[3][3];
```
int *p = &a[0][0];
for(int i=0;i<9;i++)
        p[i] = i+1;
for(int i=0;i<3;i++)
        printf(“%d”,a[2][i]);
```
结果：10 11 12 13 14

4.

```
#include<stdio.h>
#define M 5
#define N M+M
int main(void)
{
int k;
k=N*N*5;
printf("%d\n",k);
return 0;
}
```
运行结果：55

5.

```c
#include <stdio.h>
int main(void)
{
int a = 6, b = 8;
while (a--);
while (--b);
printf("a=%d b=%d\n", a, b);
return 0;
}
```

运行结果是：a=-1 b=0

6.

```c
#include <stdio.h>
int main(void)
{
int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, *p = a;
printf("x:%d y:%d\n", *a, *(p + 5));
return 0;
}
```

运行结果是：x: 1 y: 6

编程题

1. 题目：有5个学生坐在一起，问第5个学生多少岁，他说比第4个学生大2岁。问第4个学生岁数，他说比第3个学生大2岁。问第3个学生，又说比第2个学生大2岁。问第2个学生，说比第一个学生大2岁。最后问第一个学生，他说是10岁。请问第5个学生多大。

参考答案：18

```
#include<algorithm>
#include<math.h>
#include<vector>
#include<map>

using namespace std;

//递归函数的实现
int fun(int n)
{
if(n == 1)
  return 10; //第一个人年龄为10岁
else if(n>1)
  return fun(n-1)+2; //后面每个人年龄依次在前一人基础上加2
else
 return -1;  //如果你给的n值不合法（比如为负数），就返回-1
}

int main()
{
int k=fun(5);
printf("第5个人的年龄为:%d\n",k);
return 0;
}
```

编程题

2.编写一个函数，对一个无符号短整型数，取它的偶数位（即从左边起第2、4、6… 位）与奇数位（即从左边起第1 、3、5…）分别组成新的无符短整数并通过形参传回调用参数。

原型：

void Split(unsigned short a,unsigned short * pOdd,unsigned short * pEven);

其中pOdd代表奇数位，pEven代表偶数位。

```
/*巧用atoi()和itoa()函数*/
void Split(unsigned short a, unsigned short *pOdd, unsigned short
*pEven)
{
   char s[20];
   char odd[20];
   char even[20];
   itoa(a,s,10);  //将数字a转换为字符串
   int i;
   //strrev(s);
   for(i=0;i<strlen(s);i++)
   {
       if(i%2 == 0)/*偶数位*/
       {
           even[i/2] = s[i] ;
           even[i/2+1] = '\0';
       }
       else
       {
           odd[i/2] = s[i];
           odd[i/2+1]='\0';
       }
   }
   //unsigned short sodd,seven;
   *pOdd = atoi(odd);
   *pEven = atoi(even);
}

int main()
{
   unsigned short a = 12345;
    unsigned short  *pOdd = (unsigned short *)malloc(sizeof(unsigned
short));
    unsigned short *pEven = (unsigned short*)malloc(sizeof(unsigned
short));
   Split(a,pOdd,pEven);
   printf("%u\t%u\n",*pOdd,*pEven);
   return 0;
}
```

编程题

3.甲乙两队进行比赛，甲队有a、b、c三人，乙队有x、y、 z三人，有人想知道比赛对手，a说不和x比赛，c说不和x，z比赛，编程找三对赛手名单。

解：

```
#include<stdio.h>
#include<string.h>
#include<iostream>
#include<algorithm>
#include<math.h>
#include<vector>
#include<map>

using namespace std;

int main()
{
   char x = 'a',y,z;
   for(x = 'a'; (x<='c');x++)
   {
       for(y = 'a';y<='c';y++)
       {
           for(z = 'a'; z <= 'c'; z++)
           {
                 if( x!=y && x!=z && y!=z && x!='a' && x!='c' && z!=
'c' )
               {
                   printf("x -> %c\n",x);
                   printf("y -> %c\n",y);
                   printf("z -> %c\n",z);
               }

           }
       }
   }
}
```

编程题

二、1.字符串循环移动 （输入一个字符串和向左移动的位数n输出循环左移的结果）

```
#include<math.h>
#include<vector>
#include<map>

using namespace std;

int main()
{
    char str[1005],n;
    while(scanf("%s %d",&str,&n)==2)
    {
        int len=strlen(str); //len为字符长度
        n=n%len;
        for(int i=n;i<len;i++)
            printf("%c",str[i]);
        for(int i=0;i<n;i++)
            printf("%c",str[i]);
        printf("\n");
    }
    return 0;
}
```

2、输出以下的杨辉三角形（要求输出10行）（7分）

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10  10  5   1
⋮   ⋮   ⋮   ⋮   ⋮   ⋮
```

```
#include <stdio.h>
#define N  10
int main( )
{ int i,j,a[N][N];                          //数组为 10 行 10 列
  for (i=0;i<N;i++)
     {a[i][i]=1;                            //使对角线元素的值为 1
      a[i][0]=1;                            //使第 1 列元素的值为 1
     }
  for (i=2;i<N;i++)                         //从第 3 行开始处理
    for (j=1;j<=i-1;j++)
      a[i][j]=a[i-1][j-1]+a[i-1][j];
  for (i=0;i<N;i++)
    {for (j=0;j<=i;j++)
       printf("%6d",a[i][j]);               //输出数组各元素的值
     printf("\n");
    }
  printf("\n");
return 0;
}
```

3.小顶堆：

有N(N>>10000)个整数，求出其中的前K个最大的数。

为了方便测试我就输入10个数字求前4个最大的。

输入：（第一行n，k代表n个数字求k个最大的）

10 4
1 2 3 4 5 6 7 8 9 10

输出：

7 8 9 10

```
int n;
int dui[10];
int K;
```

```
void UpToDown(int i)
{
    int t1,t2,tmp,pos;
    t1=2*i; //左孩子(存在的话)
    t2=t1+1;    //右孩子(存在的话)
    if(t1>K)    //无孩子节点
        return;
    else
    {
        if(t2>K)  //只有左孩子
            pos=t1;
        else
            pos=dui[t1]>dui[t2]? t2:t1;

        if(dui[i]>dui[pos]) //pos 保存在子孩子中，数值较小者的位置
        {
            tmp=dui[i];dui[i]=dui[pos];dui[pos]=tmp;
            UpToDown(pos);
        }
    }
}

void create_dui()
{
    int i;
    int pos=K/2;       //从末尾数，第一个非叶节点的位置 K/2
    for(i=pos;i>=1;i--)
        UpToDown(i);
}

int main()
{
    int i;
    int tmp;
    while(scanf("%d %d",&n,&K)==2)
    {
        for(i=1;i<=K;i++) //先输入 K 个
```

```
            scanf("%d",&dui[i]);
        create_dui();  //建小顶堆
        for(i=K+1;i<=n;i++)
        {
            scanf("%d",&tmp);
            if(tmp>dui[1])  //只有大于根节点才处理
            {
                dui[1]=tmp;
                UpToDown(1);     //向下调整堆
            }
        }
        for(int i=1;i<=K;i++) printf("%d ",dui[i]);
        printf("\n");
    }
    return 1;
}
```

4、八皇后问题，是一个古老而著名的问题，是回溯算法的典型案例。该问题是国际西洋棋棋手马克斯·贝瑟尔于1848年提出：在8×8格的国际象棋上摆放八个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法。（9分）

```c
#include<stdio.h>

int sum = 0; //方法总数
int chess[8][8]={0}; //8*8的棋盘

//判断此次放置是否合理|合理则返回1，否则返回0
int check(int row,int col)
{
    int i,j;
```

```
    for (i = 0; i < 8; i++) //判断皇后所在列
    {
        if (chess[i][col] == 1)
            return 0;
    }
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--) //判断左下对角线
    {
        if (chess[i][j] == 1)
            return 0;
    }
    for (i = row, j = col; i >= 0 && j < 8; i--, j++) //判断右下对角线
    {
        if (chess[i][j] == 1)
            return 0;
    }
    return 1;
}
//打印输出结果
void print()
{
    int i,j;
    printf("第%d种解法\n",sum+1);
    for (i = 0; i < 8; i++)
    {
        for (j = 0; j < 8; j++)
        {
            if (chess[i][j] == 1)
                printf("Q "); //皇后放置位置输出'Q'
            else
                printf("# "); //其余位置输出'#'
        }
        printf("\n");
    }
    printf("\n");
}
//寻找解法
void search(int row)
{
    if (row == 8) //此时0-7行均已正常放置皇后
    {
        print();
        sum++;
        return;
    }
    int col; //列
    for (col = 0; col < 8; col++) //假定将皇后放置在第row行第col列
    {
        if (check(row, col)) //若可以放置在此
        {
            chess[row][col] = 1; //修改该元素值为1
            search(row + 1); //进行下一次递归
            chess[row][col] = 0; //恢复被修改的值，否则会影响后续递归
```

```
        }
    }
}

int main()
{
    search(0);
    printf("共有%d种解法\n",sum);
    return 0;
}
```

计算机专业综合测试部分

一、

题目：DeepSeek算法优化与评估

假设你正在使用DeepSeek框架开发一个图像分类模型，请回答以下问题：

1. 基础原理

DeepSeek框架中默认的损失函数为交叉熵损失（Cross-Entropy Loss），请解释：

- 为什么交叉熵损失适合多分类任务?
- 如果遇到类别不平衡问题（如正负样本比例1:9），如何修改损失函数？（参考答案：改用Focal Loss或加权交叉熵损失）

2. 训练调优

在训练过程中发现验证集准确率波动较大，可能的原因有哪些？请至少列出3种并给出对应解决方案。

（参考答案：① 学习率过高 → 使用学习率衰减；② 小批量数据噪声 → 增大batch size；③ 模型过拟合 → 增加Dropout层）

3. 性能提升

现有DeepSeek模型在测试集上的F1-score为0.72，请设计一个优化方案流程图，要求包含以下环节：

- 数据增强策略
- 模型结构改进方向
- 训练超参数调优方法

（参考答案示例：

① 数据增强：随机裁剪+Mixup

② 模型改进：引入注意力机制模块

③ 超参数：网格搜索学习率/权重衰减组合）

4.

如果你是 DeepSeek 技术的开发者，你会如何平衡技术创新与社会责任?

1. 技术透明性与可解释性

- **措施**：
  - 在开发过程中，确保算法的透明性，避免“黑箱”操作。例如，提供算法决策的逻辑解释，让用户理解结果是如何产生的。
  - 开发可解释性工具，帮助用户和监管机构理解技术的运作机制。
- **社会责任**：
  - 避免技术被滥用或误解，确保用户对技术有充分的知情权。
  - 通过透明性建立用户信任，减少对技术的恐惧和排斥。

2. 数据隐私与安全

- **措施**：
  - 采用隐私保护技术，如差分隐私、联邦学习等，确保用户数据在训练和使用过程中不被泄露。
  - 严格遵守数据保护法规（如 GDPR），确保用户数据的合法收集和使用。
- **社会责任**：
  - 保护用户隐私是技术开发者的基本责任，避免因数据泄露或滥用导致的社会问题。
  - 通过技术手段和制度设计，确保用户对自己的数据有控制权。

二、

题目 1

请详细阐述深度学习（Deep Learning）的概念，并分析它与机器学习（Machine Learning）的关系。要求至少从定义、方法、应用领域三个方面进行对比分析。

参考答案：

- **定义**：机器学习是一门多领域交叉学科，专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能。它基于数据，通过构建模型来进行预测和决策。而深度学习是机器学习的一个分支领域，它是基于对数据进行表征学习的方法 。深度学习通过构建具有很多层的神经网络模型，自动从大量数据中学习特征和模式。
- **方法**：机器学习方法包括决策树、支持向量机、朴素贝叶斯等传统算法。这些算法通常需要人工进行特征工程，即手动提取和选择对模型有意义的特征。深度学习则主要依赖深度神经网络，通过构建包含多个隐藏层的神经网络结构，让模型自动从原始数据中学习到不同层次的特征表示，减少了对人工特征工程的依赖。
- **应用领域**：机器学习在很多领域都有应用，如数据挖掘、生物信息学、金融风险评估等。在金融风险评估中，可利用逻辑回归等机器学习算法来预测贷款违约的可能性。深度学习则在图像识别（如人脸识别、自动驾驶中的交通标志识别）、语音识别（如智能语音助手）、自然语言处理（如机器翻译、文本生成）等对数据处理能力和模型复杂度要求较高的领域取得了显著成果。

题目 2

什么是神经网络（Neural Network）？请详细描述一个简单的前馈神经网络（Feedforward Neural Network）的结构，并解释各层的作用。

参考答案：

神经网络是一种模仿生物神经网络的结构和功能的数学模型或计算模型。它由大量的节点（神经元）和连接这些节点的边组成，通过对数据的学习来调整节点之间的连接权重，从而实现对数据的分类、预测、聚类等任务。

简单的前馈神经网络结构包含以下部分：

- **输入层**：是网络与外部数据的接口，负责接收输入数据。输入层的神经元数量与输入数据的特征数量一致。例如，在手写数字识别任务中，如果输入的是 28×28 像素的图像，将其展开成一维向量后有 784 个元素，那么输入层神经元数量就是 784。输入层神经元仅起到传递数据的作用，不进行任何计算。
- **隐藏层**：位于输入层和输出层之间，可以有一层或多层。隐藏层中的神经元对输入数据进行非线性变换。每个神经元会接收来自输入层（或前一层隐藏层）所有神经元的输入信号，并通过加权求和的方式计算一个综合输入值，再将这个值通过激活函数（如 ReLU、Sigmoid 等）进行非线性变换 。通过多层隐藏层的组合，网络可以学习到数据中复杂的非线性关系，提取出更高级的特征。
- **输出层**：根据隐藏层传递过来的信息，产生最终的输出结果。输出层神经元数量取决于具体任务。例如，在二分类任务中，输出层可以只有 1 个神经元，通过输出值的大小（如大于 0.5 判定为一类，小于 0.5 判定为另一类）来表示分类结果；在多分类任务中，若有 n 个类别，输出层通常有 n 个神经元，每个神经元对应一个类别，通过 softmax 函数将输出值转换为概率分布，概率最大的类别即为预测类别。

题目 3

解释卷积神经网络（Convolutional Neural Network, CNN）中的卷积层（Convolutional Layer）和池化层（Pooling Layer）的作用，并举例说明它们在图像识别任务中的应用。

参考答案：

- **卷积层作用**：卷积层主要用于提取数据的局部特征。它通过卷积核在数据上滑动进行卷积操作。卷积核是一个小的权重矩阵，其大小通常为 3×3、5×5 等。在图像识别中，当卷积核在图像上滑动时，它会对图像的局部区域进行加权求和，得到一个新的特征值。例如，一个 3×3 的卷积核在图像上每次移动一个像素，对每个 3×3 的图像区域进行计算，将图像中的每个局部区域转换为一个新的特征值，从而形成一个新的特征图。多个不同的卷积核可以同时作用于图像，每个卷积核学习到不同类型的局部特征，如边缘、纹理等。通过这种方式，卷积层能够在大幅减少参数数量的同时，有效地提取图像的特征，大大提高了模型的训练效率和泛化能力。
- **池化层作用**：池化层通常接在卷积层之后，主要用于对特征图进行下采样操作，常见的池化方法有最大池化和平均池化。在最大池化中，将特征图划分为多个不重叠的子区域，每个子区域中取最大值作为池化后的输出；平均池化则是取子区域的平均值作为输出。以 2×2 的最大池化为例，在图像识别中，将特征图按照 2×2 的窗口进行划分，每个窗口中取最大的像素值作为该窗口池化后的结果，这样可以将特征图的尺寸缩小为原来的 1/4 。池化层的作用一是减少数据维度，降低计算量，二是在一定程度上提取主要特征，同时还能防止过拟合，提高模型的泛化能力。例如在图像分类任务中，经过多层卷积和池化操作后，能够在保留图像关键特征的前提下，大大减少后续全连接层的计算量，提高模型的运行速度和准确性。

题目 4

电商平台上存在大量用户对商品的评价内容，你的任务是构建一个模型，判断这些评价中提及的商品属性（如质量、外观、功能等）是否符合用户期望，判断结果分为符合、基本符合、不符合三种情况。设计这样一个商品属性期望判断模型需要考虑哪些因素?

答案

1. **数据收集与整理（2 分）**
   - **多渠道收集**：从电商平台的商品详情页、追加评论区、问答板块等多位置收集评价数据，确保数据全面。同时，按照不同商品类别、品牌进行分类存储，以便后续分析不同类别商品评价的特点差异。
   - **去噪处理**：去除重复评价、乱码、广告信息等无效数据，对特殊符号和表情进行统一的编码转换或语义标注，提升数据的纯净度。

2. **特征提取（3 分）**
   - **关键词提取**：通过词性标注、词频统计等方法，提取与商品属性相关的关键词，如 “质量”“好看”“好用” 等，结合关键词在评价中的出现频率和位置，构建特征向量。
   - **语义特征挖掘**：利用依存句法分析、语义角色标注等技术，挖掘评价文本中词语之间的语义关系，如 “这款手机的拍照功能很强大”，分析出 “拍照功能” 与 “强大” 之间的修饰关系，作为判断功能是否符合期望的语义特征。

3. **模型选择与训练（3 分）**
   - **传统机器学习模型**：逻辑回归模型可通过对特征进行线性组合，预测商品属性是否符合期望，训练过程中调整正则化参数，防止过拟合。决策树模型能根据特征进行规则划分，构建直观的决策规则，训练时可通过剪枝操作优化模型。
   - **深度学习模型**：Transformer 模型通过自注意力机制，能更好地捕捉文本中的长距离依赖关系，对评价文本中的语义理解更准确。在训练时，设置合适的层数、头数、隐藏层维度等超参数，利用预训练语言模型（如 BERT）初始化模型参数，加快训练收敛速度，同时采用 Dropout、层归一化等技术防止过拟合。

题目 5

题目

在自然语言处理领域，深度学习模型在机器翻译任务中发挥着关键作用。假设你要构建一个基于 Transformer 架构的深度学习模型，实现英语到中文的翻译。请回答以下问题：

1. 简述 Transformer 架构中多头注意力机制（Multi - Head Attention）的工作原理及其在英译汉任务中的优势。（4 分）

答案

1. **多头注意力机制工作原理及优势**：
   - **工作原理**：多头注意力机制将输入的查询（Query）、键（Key）和值（Value）分别通过多个不同的线性变换，得到多个不同的子查询、子键和子值。然后在每个子空间中分别计算注意力分数，即通过点积计算查询与键的相似度，经过 Softmax 归一化后得到注意力权重，再用注意力权重与值进行加权求和，得到每个子空间的输出。最后将多个子空间的输出拼接起来，再经过一次线性变换得到最终的输出。（2 分）
   - **在英译汉任务中的优势**：多头注意力机制可以并行地从不同表示子空间中学习到不同的注意力信息，能够同时关注输入文本的不同部分，捕捉更丰富的语义和句法关系。例如在翻译复杂句子时，不同的头可以分别关注句子的主语、谓语、宾语等不同成分之间的关系，从而更好地理解英语原文的含义，提高翻译的准确性和流畅性。（2 分）

三、英译汉

In the realm of computer vision, deep learning has brought about revolutionary changes. Images, being a rich source of visual information, are composed of pixels arranged in a two - dimensional or multi - dimensional structure. Analyzing and understanding these complex visual patterns to extract meaningful insights has long been a goal in the field. Traditional methods for image recognition, such as using hand - crafted features like SIFT (Scale - Invariant Feature Transform) and HOG (Histogram of Oriented Gradients), have limitations in handling the vast diversity of images.

With the advent of deep learning, convolutional neural networks (CNNs) have emerged as a dominant approach. CNNs are designed to automatically learn hierarchical features from images through convolutional layers, pooling layers, and fully - connected layers. Over the years, researchers have been constantly improving CNN architectures. For example, the AlexNet, the first successful large - scale CNN, demonstrated the power of deep neural networks in image classification. Subsequently, more advanced architectures like VGGNet, ResNet, and Inception have been proposed. ResNet, in particular, introduced the concept of residual connections, which significantly alleviated the problem of vanishing gradients during training, enabling the training of much deeper neural networks.

Recently, the focus has shifted towards exploring the application of CNNs in more complex tasks, such as object detection in real - time scenarios, semantic segmentation for autonomous driving, and image generation using generative adversarial networks (GANs). As the amount of available image data continues to grow exponentially, and the computing power of GPUs (Graphics Processing Units) becomes more powerful, the potential of deep - learning - based computer vision techniques seems limitless. However, challenges still remain, such as dealing with limited data in certain specialized domains, and ensuring the interpretability of the models, especially in safety - critical applications.

答案

在计算机视觉领域，深度学习带来了革命性的变化。图像作为丰富的视觉信息来源，由以二维或多维结构排列的像素组成。分析和理解这些复杂的视觉模式以提取有意义的信息，长期以来一直是该领域的目标。传统的图像识别方法，如使用尺度不变特征变换（SIFT）和方向梯度直方图（HOG）等手工特征，在处理种类繁多的图像时存在局限性。

随着深度学习的出现，卷积神经网络（CNN）已成为主流方法。CNN 旨在通过卷积层、池化层和全连接层，从图像中自动学习层次化特征。多年来，研究人员不断改进 CNN 架构。例如，首个成功的大规模 CNN——AlexNet，展示了深度神经网络在图像分类中的强大能力。随后，诸如 VGGNet、ResNet 和 Inception 等更先进的架构相继问世。特别是 ResNet 引入了残差连接的概念，显著缓解了训练过程中的梯度消失问题，使得训练更深层次的神经网络成为可能。

最近，研究重点转向探索 CNN 在更复杂任务中的应用，比如实时场景中的目标检测、自动驾驶中的语义分割，以及使用生成对抗网络（GAN）进行图像生成。随着可用图像数据量呈指数级增长，并且图形处理单元（GPU）的计算能力变得更加强大，基于深度学习的计算机视觉技术潜力似乎无穷无尽。然而，挑战依然存在，比如在某些特定领域处理数据有限的问题，以及确保模型的可解释性，尤其是在安全关键型应用中。