```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// 定义学生集合S，习题集合E和知识点集合K
int S[10]; // 学生集合
int E[10]; // 习题集合
int K[10]; // 知识点集合

// 生成表示学生知识掌握程度的向量hsi
void generate_hsi(int *hsi, int student) {
    // 根据学生的知识掌握程度生成向量hsi
}
```

```c
// 生成表示习题难度和区分度的向量dei
void generate_dei(int *dei, int exercise) {
    // 根据习题的难度和区分度生成向量dei
}

// 生成习题-知识点关联向量qi
void generate_qi(int *qi, int exercise) {
    // 根据习题-知识点关联矩阵Q生成向量qi
}

// 使用神经网络模型学习学生向量、习题向量、习题-知识点关联向量之间的交互
double neural_network_model(int *hsi, int *dei, int *qi) {
    // 使用神经网络模型（如NCD）学习学生向量、习题向量、习题-知识点关联向量之间的交互
    // 并输出正确回答问题的概率
}
```

```c
// 引入注意力机制，计算不同部分的权重
void attention_mechanism(int *weights, int *hsi, int *dei, int *qi) {
    // 引入注意力机制，通过计算不同部分的权重，增强模型对学生实际掌握程度的敏感性和判断的合理性
}

// 训练模型，并在验证集和测试集上评估模型性能
void train_and_evaluate() {
    // 训练模型，并在验证集和测试集上评估模型性能，记录准确率、均方根误差和曲线下面积等指标
}

int main() {
    // 初始化学生集合S、习题集合E和知识点集合K
    for (int i = 0; i < 10; i++) {
        S[i] = i + 1;
        E[i] = i + 1;
        K[i] = i + 1;
    }
```

```
    // 对于每个学生si∈S，生成一个表示其知识掌握程度的向量hsi
    for (int i = 0; i < 10; i++) {
        int hsi[10];
        generate_hsi(hsi, S[i]);
    }

    // 对于每个习题ei∈E，生成一个表示其难度和区分度的向量dei
    for (int i = 0; i < 10; i++) {
        int dei[10];
        generate_dei(dei, E[i]);
    }

    // 根据习题-知识点关联矩阵Q，生成习题-知识点关联向量qi
    for (int i = 0; i < 10; i++) {
        int qi[10];
        generate_qi(qi, E[i]);
    }
```

```
    // 使用神经网络模型学习学生向量、习题向量、习题-知识点关联向量之间的交互
    for (int i = 0; i < 10; i++) {
        int hsi[10];
        generate_hsi(hsi, S[i]);
        int dei[10];
        generate_dei(dei, E[i]);
        int qi[10];
        generate_qi(qi, E[i]);
        double probability = neural_network_model(hsi, dei, qi);
        printf("学生 %d 正确回答问题的概率为：%f\n", S[i], probability);
    }
```

```
    // 引入注意力机制，计算不同部分的权重
    for (int i = 0; i < 10; i++) {
        int hsi[10];
        generate_hsi(hsi, S[i]);
        int dei[10];
        generate_dei(dei, E[i]);
        int qi[10];
        generate_qi(qi, E[i]);
        int weights[10];
        attention_mechanism(weights, hsi, dei, qi);
    }

    // 训练模型，并在验证集和测试集上评估模型性能
    train_and_evaluate();

    return 0;
}
```