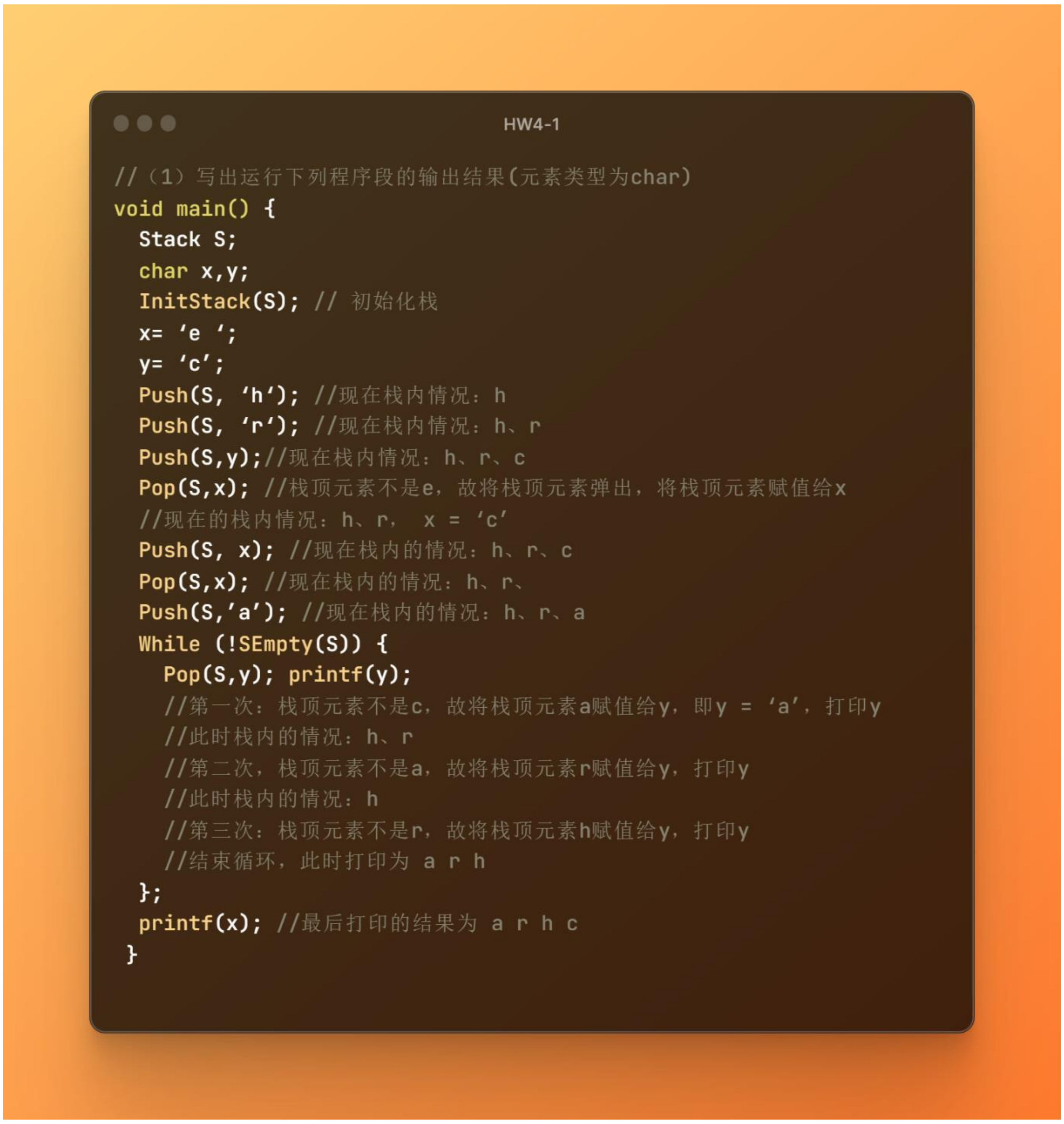
第一题：

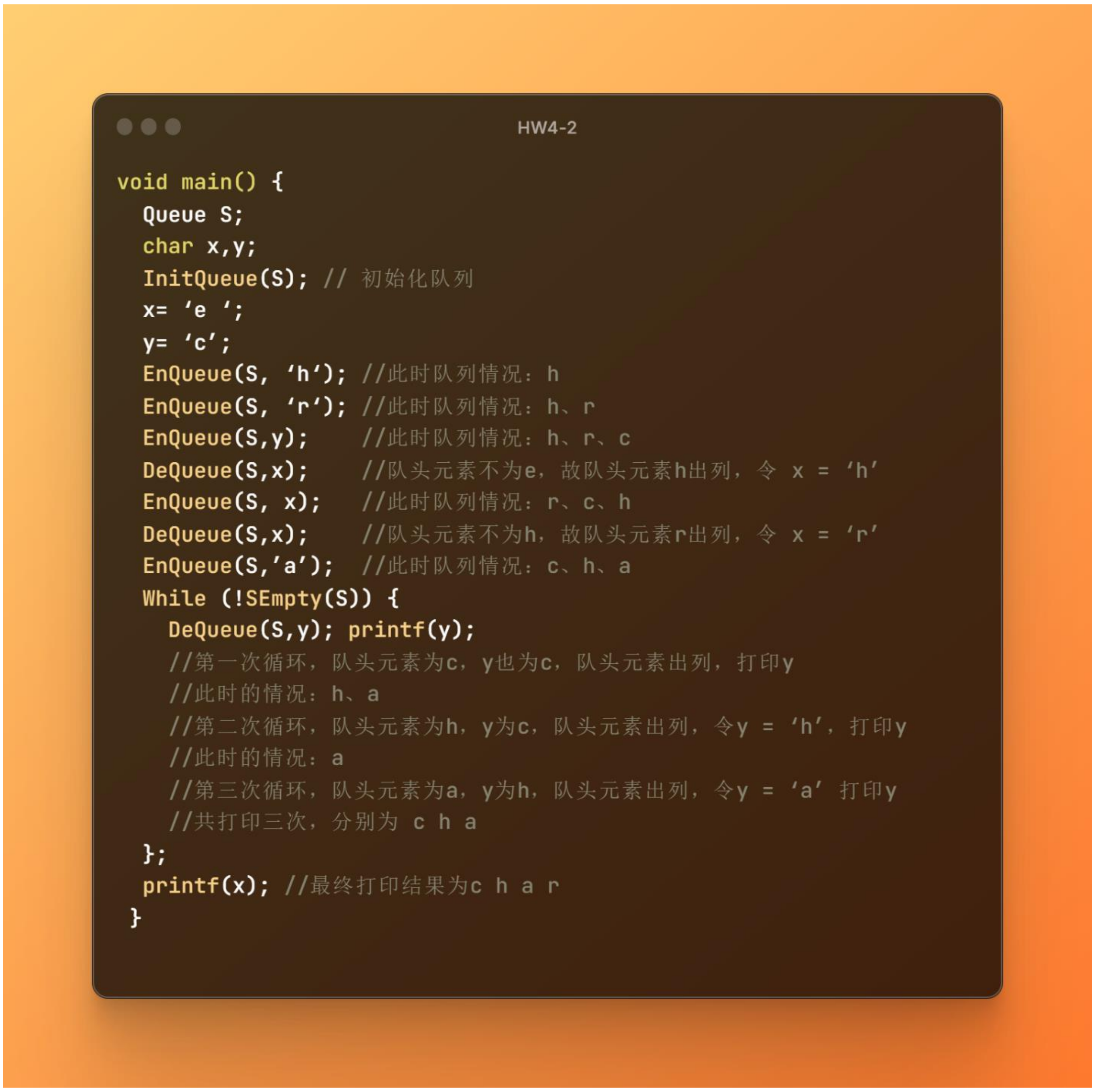
（1）代码（最后的打印结果为 arhc）



HW4-1

```
//（1）写出运行下列程序段的输出结果(元素类型为char)
void main() {
  Stack S;
  char x,y;
  InitStack(S); // 初始化栈
  x= ‘e ’;
  y= ‘c’;
  Push(S, ‘h‘); //现在栈内情况：h
  Push(S, ‘r‘); //现在栈内情况：h、r
  Push(S,y);//现在栈内情况：h、r、c
  Pop(S,x); //栈顶元素不是e，故将栈顶元素弹出，将栈顶元素赋值给x
  //现在的栈内情况：h、r，  x = ‘c’
  Push(S, x); //现在栈内的情况：h、r、c
  Pop(S,x); //现在栈内的情况：h、r、
  Push(S,’a’); //现在栈内的情况：h、r、a
  While (!SEmpty(S)) {
    Pop(S,y); printf(y);
    //第一次：栈顶元素不是c，故将栈顶元素a赋值给y，即y = ‘a’，打印y
    //此时栈内的情况：h、r
    //第二次，栈顶元素不是a，故将栈顶元素r赋值给y，打印y
    //此时栈内的情况：h
    //第三次：栈顶元素不是r，故将栈顶元素h赋值给y，打印y
    //结束循环，此时打印为 a r h
  };
  printf(x); //最后打印的结果为 a r h c
 }
```

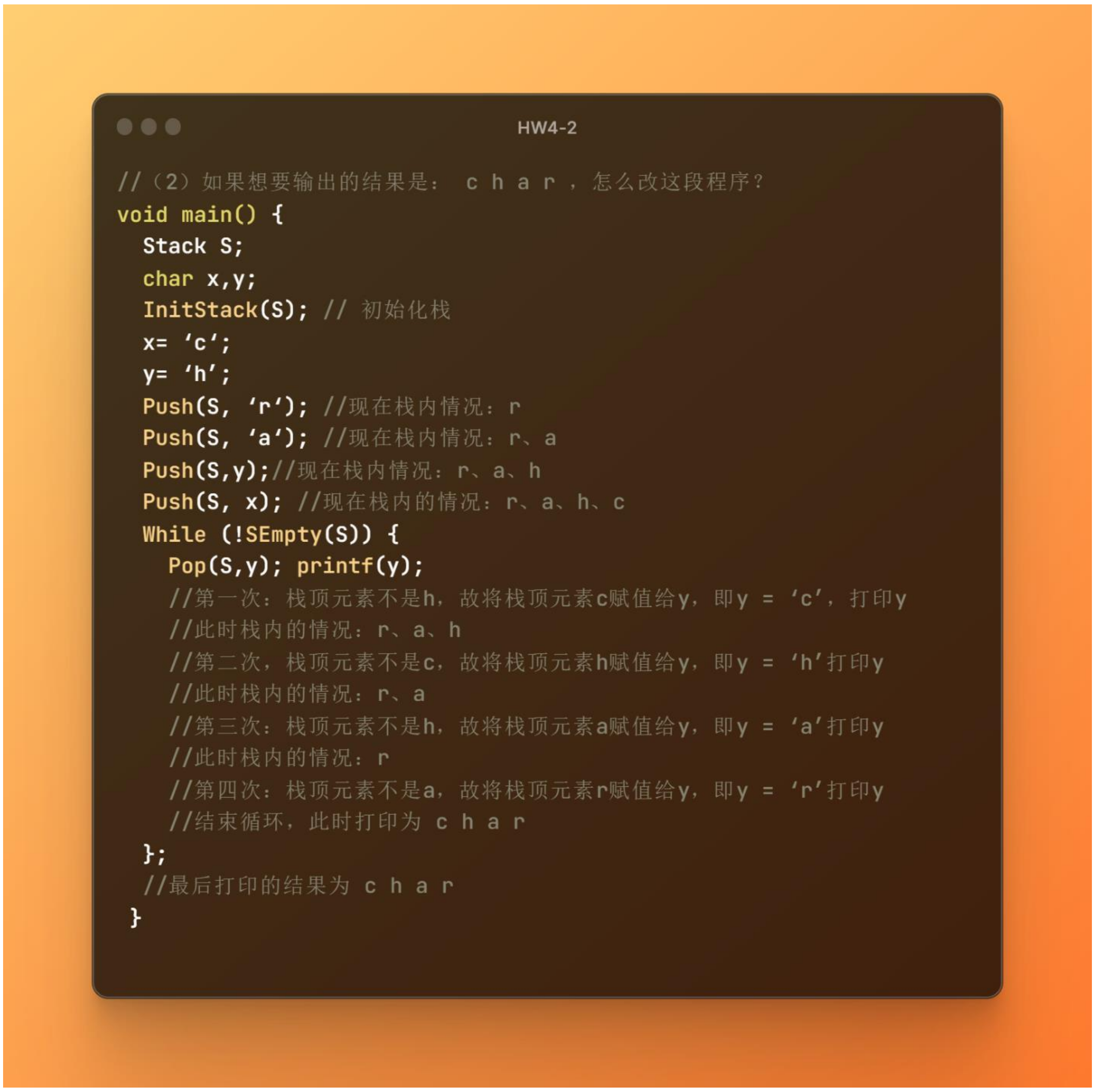
（2）代码（最后打印结果）



```
void main() {
  Queue S;
  char x,y;
  InitQueue(S); // 初始化队列
  x= ‘e ’;
  y= ‘c’;
  EnQueue(S, ‘h’); //此时队列情况：h
  EnQueue(S, ‘r’); //此时队列情况：h、r
  EnQueue(S,y);    //此时队列情况：h、r、c
  DeQueue(S,x);    //队头元素不为e，故队头元素h出列，令 x = ‘h’
  EnQueue(S, x);   //此时队列情况：r、c、h
  DeQueue(S,x);    //队头元素不为h，故队头元素r出列，令 x = ‘r’
  EnQueue(S,’a’);  //此时队列情况：c、h、a
  While (!SEmpty(S)) {
    DeQueue(S,y); printf(y);
    //第一次循环，队头元素为c，y也为c，队头元素出列，打印y
    //此时的情况：h、a
    //第二次循环，队头元素为h，y为c，队头元素出列，令y = ‘h’，打印y
    //此时的情况：a
    //第三次循环，队头元素为a，y为h，队头元素出列，令y = ‘a’ 打印y
    //共打印三次，分别为 c h a
  };
  printf(x); //最终打印结果为c h a r
 }
```

第二题：

代码：

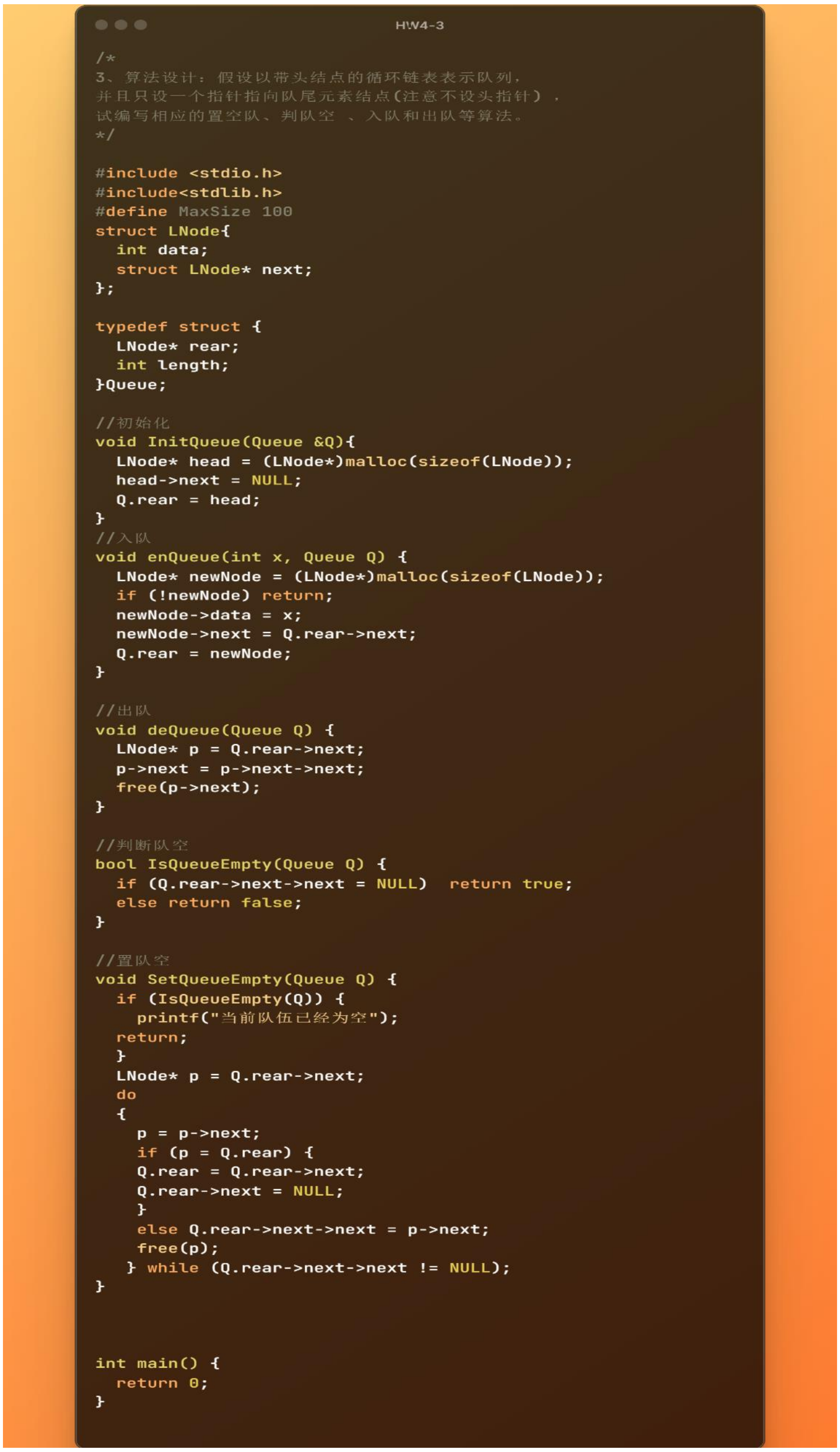


HW4-2

```
//（2）如果想要输出的结果是： c h a r ，怎么改这段程序？
void main() {
  Stack S;
  char x,y;
  InitStack(S); // 初始化栈
  x= ‘c’;
  y= ‘h’;
  Push(S, ‘r’); //现在栈内情况：r
  Push(S, ‘a’); //现在栈内情况：r、a
  Push(S,y);//现在栈内情况：r、a、h
  Push(S, x); //现在栈内的情况：r、a、h、c
  While (!SEmpty(S)) {
    Pop(S,y); printf(y);
    //第一次：栈顶元素不是h，故将栈顶元素c赋值给y，即y = ‘c’，打印y
    //此时栈内的情况：r、a、h
    //第二次，栈顶元素不是c，故将栈顶元素h赋值给y，即y = ‘h’打印y
    //此时栈内的情况：r、a
    //第三次：栈顶元素不是h，故将栈顶元素a赋值给y，即y = ‘a’打印y
    //此时栈内的情况：r
    //第四次：栈顶元素不是a，故将栈顶元素r赋值给y，即y = ‘r’打印y
    //结束循环，此时打印为 c h a r
  };
  //最后打印的结果为 c h a r
 }
```

第三题：（核心思想，尾指针 rear->next 指向的是头节点）

代码：

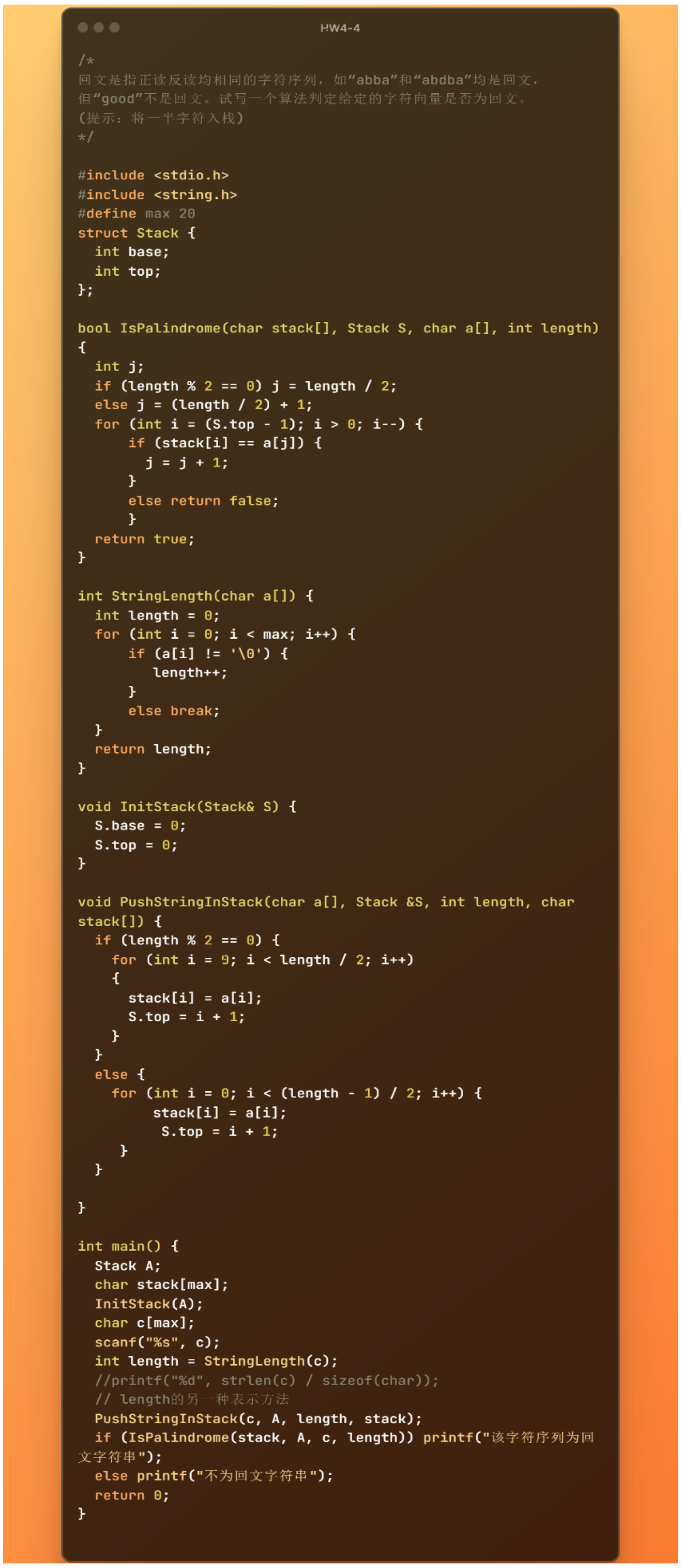


HW4-3

```c
/*
3、算法设计：假设以带头结点的循环链表表示队列，
并且只设一个指针指向队尾元素结点(注意不设头指针) ，
试编写相应的置空队、判队空 、入队和出队等算法。
*/

#include <stdio.h>
#include<stdlib.h>
#define MaxSize 100
struct LNode{
  int data;
  struct LNode* next;
};

typedef struct {
  LNode* rear;
  int length;
}Queue;

//初始化
void InitQueue(Queue &Q){
  LNode* head = (LNode*)malloc(sizeof(LNode));
  head->next = NULL;
  Q.rear = head;
}
//入队
void enQueue(int x, Queue Q) {
  LNode* newNode = (LNode*)malloc(sizeof(LNode));
  if (!newNode) return;
  newNode->data = x;
  newNode->next = Q.rear->next;
  Q.rear = newNode;
}

//出队
void deQueue(Queue Q) {
  LNode* p = Q.rear->next;
  p->next = p->next->next;
  free(p->next);
}

//判断队空
bool IsQueueEmpty(Queue Q) {
  if (Q.rear->next->next = NULL)  return true;
  else return false;
}

//置队空
void SetQueueEmpty(Queue Q) {
  if (IsQueueEmpty(Q)) {
    printf("当前队伍已经为空");
  return;
  }
  LNode* p = Q.rear->next;
  do
  {
    p = p->next;
    if (p = Q.rear) {
    Q.rear = Q.rear->next;
    Q.rear->next = NULL;
    }
    else Q.rear->next->next = p->next;
    free(p);
   } while (Q.rear->next->next != NULL);
}


int main() {
  return 0;
}
```

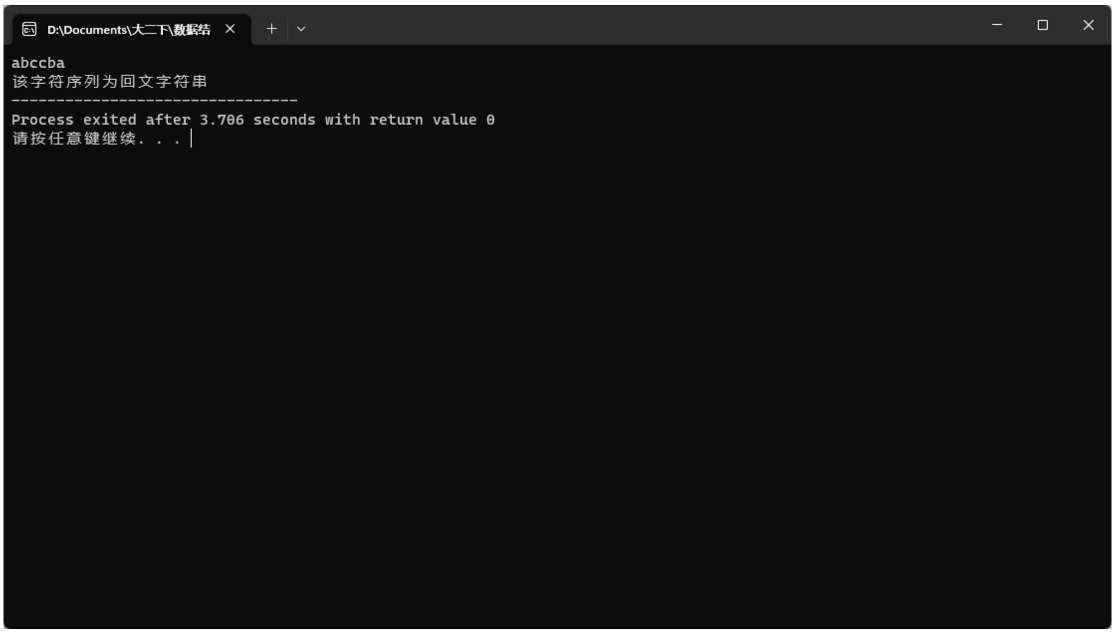
第四题：（可执行程序 HW4.4.cpp 文件）

代码：

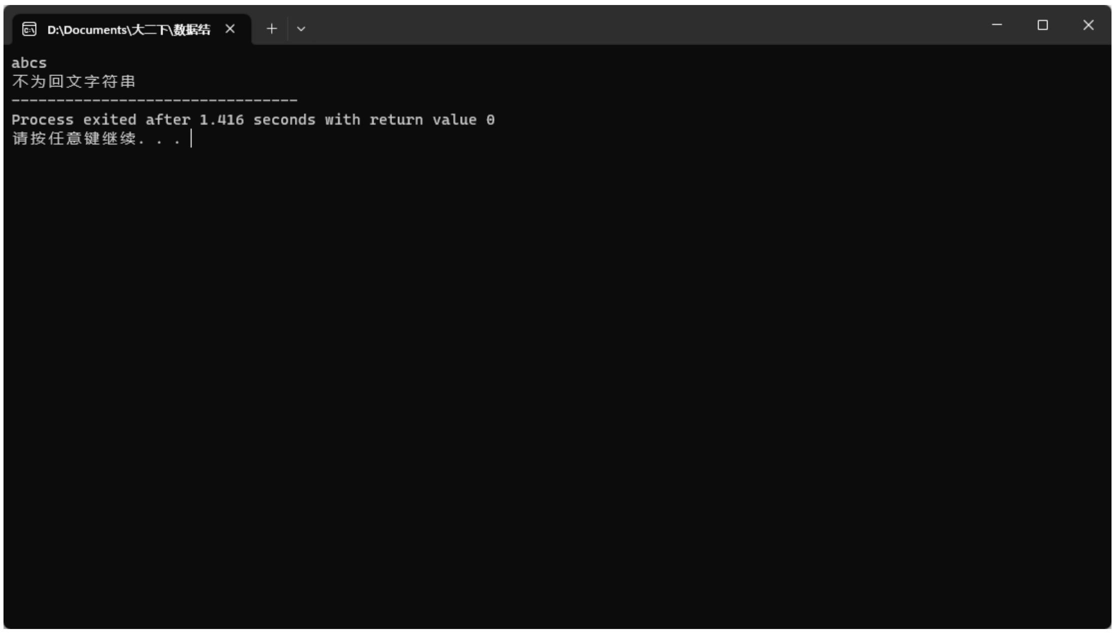


```cpp
/*
回文是指正读反读均相同的字符序列，如“abba”和“abdba”均是回文，
但“good”不是回文。试写一个算法判定给定的字符向量是否为回文。
(提示：将一半字符入栈)
*/

#include <stdio.h>
#include <string.h>
#define max 20
struct Stack {
  int base;
  int top;
};

bool IsPalindrome(char stack[], Stack S, char a[], int length)
{
  int j;
  if (length % 2 == 0) j = length / 2;
  else j = (length / 2) + 1;
  for (int i = (S.top - 1); i > 0; i--) {
      if (stack[i] == a[j]) {
        j = j + 1;
      }
      else return false;
      }
  return true;
}

int StringLength(char a[]) {
  int length = 0;
  for (int i = 0; i < max; i++) {
      if (a[i] != '\0') {
         length++;
      }
      else break;
  }
  return length;
}

void InitStack(Stack& S) {
  S.base = 0;
  S.top = 0;
}

void PushStringInStack(char a[], Stack &S, int length, char
stack[]) {
  if (length % 2 == 0) {
    for (int i = 9; i < length / 2; i++)
    {
      stack[i] = a[i];
      S.top = i + 1;
    }
  }
  else {
    for (int i = 0; i < (length - 1) / 2; i++) {
         stack[i] = a[i];
          S.top = i + 1;
     }
  }

}

int main() {
  Stack A;
  char stack[max];
  InitStack(A);
  char c[max];
  scanf("%s", c);
  int length = StringLength(c);
  //printf("%d", strlen(c) / sizeof(char));
  // length的另一种表示方法
  PushStringInStack(c, A, length, stack);
  if (IsPalindrome(stack, A, c, length)) printf("该字符序列为回
文字符串");
  else printf("不为回文字符串");
  return 0;
}
```

演示图片：



```
abccba
该字符序列为回文字符串
--------------------------------
Process exited after 3.706 seconds with return value 0
请按任意键继续. . .
```



```
abcs
不为回文字符串
--------------------------------
Process exited after 1.416 seconds with return value 0
请按任意键继续. . .
```