1. 编写一个程序，当用户输入一行英文句子后，该程序通过调用函数，找出该句子中最长的单词并输出。

源代码:

|  |
| --- |
| #include<stdio.h>  #include<string.h>  void find\_longest\_word(char sentence[]);  int main(){  char sentence[65536];  fgets(sentence,65535,stdin);  find\_longest\_word(sentence);  return 0;  }  void find\_longest\_word(char sentence[]){  char word[256][65536];  int no=0,destion=0;  for(int i=0;i<=255;i++){  while(true){  if(sentence[i]==32){  no++;  destion=0;  continue;  }  word[no][destion]=sentence[i];  i++;  }  }  for(int i=0;i<no;i++){  int now=0,longest=0,longest\_no=0;  now=strlen(word[i]);  if(now>=longest){  longest=now;  longest\_no=i;  }  printf("%s",word[longest\_no]);  }  } |

输出:

|  |
| --- |
| Segmentation fault |

1. 编写一个程序，用户输入6个学生的学号和姓名，该程序通过调用函数，将学号按照从小到大顺序排序，姓名顺序也随之调整，并输出调整顺序后的学号和姓名。

源代码:

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  #include<string.h>  #define NUM 6  void reorder(char ids[NUM][15],char names[NUM][100], int n);  int main(void)  {  char ids[NUM][15];  char names[NUM][100];  int i;  for (i = 0; i < NUM; i++) {  scanf("%s %s", ids[i], names[i]);  }  reorder(ids, names, NUM);  return 0;  }  void reorder(char ids[NUM][15],char names[NUM][100], int n)  {  int i,j,h=1;  double e,b[NUM];  char a[NUM][100];  for ( i = 0; i < NUM; i++)  {  b[i]=atoi(ids[i]);  }  for( i = 0; i < NUM-1; i++)  {  for( j = 0; j < NUM-1-i; j++)  {  if(b[j]>b[j+1])  {  e=b[j];  b[j]=b[j+1];  b[j+1]=e;  strcpy(a[i],names[j]);  strcpy(names[j],names[j+1]);  strcpy(names[j+1],a[i]);  }  }  }  printf("\n");  for (i=0;i<NUM;i++)  {  printf("%.0f %s\n",b[i],names[i]);  }    return;  } |

输出:

|  |
| --- |
| 2019010104 第一个人  2019010031 第二个人  2019010001 第三个人  2019010506 第四个人  2019010305 第五个人  2019010021 第六个人  2019010001 第三个人  2019010021 第六个人  2019010031 第二个人  2019010104 第一个人  2019010305 第五个人  2019010506 第四个人 |

1. 编写一个程序，当用户输入一个正整数后，使用递归法计算该整数的各位数字求和。

源代码:

|  |
| --- |
| #include <stdio.h>  int digit\_sum(int n);  int main(void)  {  int n;  int sum = 0;  scanf("%d", &n);  sum = digit\_sum(n);  printf("result = %d\n", sum);  return 0;  }  int digit\_sum(int n)  {  int res;  res = 0;  if (n >= 10)  {  res = digit\_sum(n % 10) + digit\_sum(n / 10);  }  else  {  res = res + n % 10;  }  return res;  } |

输出:

|  |
| --- |
| 34246  result = 19 |