部分IT公司笔试算法题

1、将一整数逆序后放入一数组中（要求递归实现）  
void convert(int \*result, int n) {  
 if(n>=10)  
  convert(result+1, n/10);  
 \*result = n%10;   
}  
int main(int argc, char\* argv[]) {  
 int n = 123456789, result[20]={};  
 convert(result, n);  
 printf("%d:", n);  
 for(int i=0; i<9; i++)  
  printf("%d", result*);  
}*

*2、求高于平均分的学生学号及成绩（学号和成绩人工输入）  
double find(int total, int n) {  
 int number, score,  average;  
 scanf("%d", &number);  
 if(number != 0) {  
  scanf("%d", &score);  
  average = find(total+score, n+1);  
  if(score >= average)  
   printf("%d:%d\n", number, score);  
  return average;  
 } else {  
  printf("Average=%d\n", total/n);  
  return total/n;  
 }  
}  
int main(int argc, char\* argv[]) {  
 find(0, 0);  
}*

*3、递归实现回文判断（如：abcdedbca就是回文，判断一个面试者对递归理解的简单程序）  
int find(char \*str, int n) {  
 if(n<=1) return 1;  
 else if(str[0]==str[n-1]) return find(str+1, n-2);  
 else  return 0;  
}*

*int main(int argc, char\* argv[]) {  
 char \*str = "abcdedcba";  
 printf("%s: %s\n", str, find(str, strlen(str)) ? "Yes" : "No");  
}*

*4、组合问题（从M个不同字符中任取N个字符的所有组合）  
void find(char \*source, char \*result, int n) {  
 if(n==1) {  
  while(\*source)  
     printf("%s%c\n", result, \*source++);  
 } else {  
  int i, j;  
  for(i=0; source != 0; i++);  
  for(j=0; result[j] != 0; j++);  
  for(; i>=n; i--) {  
   result[j] = \*source++;  
   result[j+1] = '\0';  
   find(source, result, n-1);  
  }  
 }  
}*

*int main(int argc, char\* argv[]) {  
 int const n = 3;  
 char \*source = "ABCDE", result[n+1] = {0};  
 if(n>0 && strlen(source)>0 && n<=strlen(source))  
  find(source, result, 3);  
}*

*5、分解成质因数(如435234=251\*17\*17\*3\*2，据说是华为笔试题)  
void prim(int m, int n) {  
 if(m>n) {  
  while(m%n != 0) n++;  
  m /= n;  
  prim(m, n);  
  printf("%d\*", n);  
 }  
}  
int main(int argc, char\* argv[]) {  
 int n = 435234;  
 printf("%d=", n);  
 prim(n, 2);  
}  
6、寻找迷宫的一条出路，o：通路； X：障碍。（大家经常谈到的一个小算法题）  
#define MAX\_SIZE  8  
int H[4] = {0, 1, 0, -1};   
int V[4] = {-1, 0, 1, 0};             
char Maze[MAX\_SIZE][MAX\_SIZE] = {{'X','X','X','X','X','X','X','X'},  
                                 {'o','o','o','o','o','X','X','X'},  
                                 {'X','o','X','X','o','o','o','X'},  
                             {'X','o','X','X','o','X','X','o'},  
                         {'X','o','X','X','X','X','X','X'},  
{'X','o','X','X','o','o','o','X'},  
         {'X','o','o','o','o','X','o','o'},  
                                 {'X','X','X','X','X','X','X','X'}};  
void FindPath(int X, int Y) {  
    if(X == MAX\_SIZE || Y == MAX\_SIZE) {  
       for(int i = 0; i < MAX\_SIZE; i++)  
for(int j = 0; j < MAX\_SIZE; j++)  
                  printf("%c%c", Maze[j], j < MAX\_SIZE-1 ? ' ' : '\n');  
}else for(int k = 0; k < 4; k++)   
if(X >= 0 && Y >= 0 && Y < MAX\_SIZE && X < MAX\_SIZE && 'o' == Maze[X][Y]) {  
                   Maze[X][Y] = ' ';  
                   FindPath(X+V[k], Y+H[k]);  
                   Maze[X][Y] ='o';   
}  
}  
int main(int argc, char\* argv[]) {  
    FindPath(1,0);  
}*

*7、随机分配座位，共50个学生，使学号相邻的同学座位不能相邻(早些时候用C#写的，没有用C改写）。  
static void Main(string[] args)  
{  
 int Tmp = 0, Count = 50;     
 int[] Seats = new int[Count];     
 bool[] Students = new bool[Count];  
 System.Random RandStudent=new System.Random();  
 Students[Seats[0]=RandStudent.Next(0,Count)]=true;  
 for(int i = 1; i < Count; ) {  
     Tmp=(int)RandStudent.Next(0,Count);  
     if((!Students[Tmp])&&(Seats[i-1]-Tmp!=1) && (Seats[i-1] - Tmp) != -1) {  
   Seats[i++] = Tmp;  
Students[Tmp] = true;  
  }  
 }  
 foreach(int Student in Seats)  
     System.Console.Write(Student + " ");  
 System.Console.Read();  
}*

*8、求网格中的黑点分布。现有6\*7的网格，在某些格子中有黑点，已知各行与各列中有黑点的点数之和，请在这张网格中画出黑点的位置。（这是一网友提出的题目，说是他笔试时遇到算法题）  
#define ROWS 6  
#define COLS 7  
int iPointsR[ROWS] = {2, 0, 4, 3, 4, 0};           // 各行黑点数和的情况  
int iPointsC[COLS] = {4, 1, 2, 2, 1, 2, 1};        // 各列黑点数和的情况  
int iCount, iFound;  
int iSumR[ROWS], iSumC[COLS], Grid[ROWS][COLS];*

*int Set(int iRowNo) {  
if(iRowNo == ROWS) {   
        for(int iColNo=0; iColNo < COLS && iSumC[iColNo]==iPointsC[iColNo]; iColNo++)   
           if(iColNo == COLS-1) {  
               printf("\nNo.%d:\n", ++iCount);   
               for(int i=0; i < ROWS; i++)  
                  for(int j=0; j < COLS; j++)  
                      printf("%d%c", Grid[j], (j+1) % COLS ? ' ' : '\n');  
               iFound = 1; // iFound = 1，有解  
           }  
    } else {  
        for(int iColNo=0; iColNo < COLS; iColNo++) {  
            if(iPointsR[iRowNo] == 0) {   
                Set(iRowNo + 1);  
   } else if(Grid[iRowNo][iColNo]==0) {   
Grid[iRowNo][iColNo] = 1;   
iSumR[iRowNo]++; iSumC[iColNo]++;                                  if(iSumR[iRowNo]<iPointsR[iRowNo] && iSumC[iColNo]<=iPointsC[iColNo])  
                     Set(iRowNo);  
else if(iSumR[iRowNo]==iPointsR[iRowNo] && iRowNo < ROWS)  
                     Set(iRowNo + 1);  
                Grid[iRowNo][iColNo] = 0;  
                iSumR[iRowNo]--;   
iSumC[iColNo]--;  
            }  
        }  
    }  
return iFound;     // 用于判断是否有解  
}  
int main(int argc, char\* argv[]) {  
    if(!Set(0))  
        printf("Failure!");   
}*

*9、有4种面值的邮票很多枚，这4种邮票面值分别1, 4, 12, 21，现从多张中最多任取5张进行组合，求取出这些邮票的最大连续组合值。（据说是华为2003年校园招聘笔试题）  
#define N 5  
#define M 5  
int k, Found, Flag[N];  
int Stamp[M] = {0, 1, 4, 12, 21};*

*// 在剩余张数n中组合出面值和Value  
int Combine(int n, int Value) {  
 if(n >= 0 && Value == 0) {  
  Found = 1;  
  int Sum = 0;  
  for(int i=0; i<N && Flag != 0; i++) {  
   Sum += Stamp[Flag];  
   printf("%d ", Stamp[Flag]);  
  }  
  printf("\tSum=%d\n\n", Sum);  
 }else for(int i=1; i<M && !Found && n>0; i++)  
  if(Value-Stamp >= 0) {  
   Flag[k++] = i;  
   Combine(n-1, Value-Stamp);  
   Flag[--k] = 0;  
  }  
 return Found;  
}*

*int main(int argc, char\* argv[]) {  
 for(int i=1; Combine(N, i); i++, Found=0);  
}*

*10、大整数数相乘的问题。（这是2002年在一考研班上遇到的算法题）  
void Multiple(char A[], char B[], char C[]) {  
    int TMP, In=0, LenA=-1, LenB=-1;  
    while(A[++LenA] != '\0');  
    while(B[++LenB] != '\0');  
    int Index, Start = LenA + LenB - 1;  
    for(int i=LenB-1; i>=0; i--) {  
        Index = Start--;  
        if(B != '0') {  
            for(int In=0, j=LenA-1; j>=0; j--) {  
                TMP = (C[Index]-'0') + (A[j]-'0') \* (B - '0') + In;  
                C[Index--] = TMP % 10 + '0';  
                In = TMP / 10;  
            }  
            C[Index] = In + '0';  
        }  
    }  
}*

*int main(int argc, char\* argv[]) {  
    char A[] = "21839244444444448880088888889";  
    char B[] = "38888888888899999999999999988";  
char C[sizeof(A) + sizeof(B) - 1];*

*for(int k=0; k<sizeof(C); k++)  
        C[k] = '0';  
    C[sizeof(C)-1] = '\0';*

*Multiple(A, B, C);  
    for(int i=0; C != '\0'; i++)  
        printf("%c", C);  
}*

*11、求最大连续递增数字串（如“ads3sl456789DF3456ld345AA”中的“456789”）  
int GetSubString(char \*strSource, char \*strResult) {  
    int iTmp=0, iHead=0, iMax=0;  
    for(int Index=0, iLen=0; strSource[Index]; Index++) {  
        if(strSource[Index] >= '0' && strSource[Index] <= '9' &&   
strSource[Index-1] > '0' && strSource[Index] == strSource[Index-1]+1) {  
            iLen++;                       // 连续数字的长度增1   
        } else {                          // 出现字符或不连续数字  
            if(iLen > iMax) {  
            iMax = iLen;  iHead = iTmp;   
            }          
        // 该字符是数字，但数字不连续  
            if(strSource[Index] >= '0' && strSource[Index] <= '9') {   
                iTmp = Index;   
iLen = 1;   
            }  
        }      
    }  
    for(iTmp=0 ; iTmp < iMax; iTmp++) // 将原字符串中最长的连续数字串赋值给结果串  
        strResult[iTmp] = strSource[iHead++];  
    strResult[iTmp]='\0';  
    return iMax;     // 返回连续数字的最大长度  
}  
int main(int argc, char\* argv[]) {  
    char strSource[]="ads3sl456789DF3456ld345AA", char strResult[sizeof(strSource)];  
printf("Len=%d, strResult=%s \nstrSource=%s\n",   
GetSubString(strSource, strResult), strResult, strSource);  
}*

*12、四个工人，四个任务，每个人做不同的任务需要的时间不同，求任务分配的最优方案。（2005年5月29日全国计算机软件资格水平考试——软件设计师的算法题）。  
#include "stdafx.h"  
#define N 4  
int Cost[N][N] = { {2, 12, 5, 32},  // 行号：任务序号，列号：工人序号  
                    {8, 15, 7, 11},  // 每行元素值表示这个任务由不同工人完成所需要的时间  
                    {24, 18, 9, 6},  
                    {21, 1, 8, 28}};  
int MinCost=1000;  
int Task[N], TempTask[N], Worker[N];  
void Assign(int k, int cost) {  
 if(k == N) {  
  MinCost = cost;   
  for(int i=0; i<N; i++)  
   TempTask = Task;  
 } else {  
  for(int i=0; i<N; i++) {   
   if(Worker==0 && cost+Cost[k] < MinCost) { // 为提高效率而进行剪枝  
    Worker = 1; Task[k] = i;   
    Assign(k+1, cost+Cost[k]);   
    Worker = 0; Task[k] = 0;  
   }  
  }  
 }  
}  
int main(int argc, char\* argv[]) {  
 Assign(0, 0);  
 printf("最佳方案总费用=%d\n", MinCost);  
 for(int i=0; i<N; i++)  /\* 输出最佳方案 \*/  
  printf("\t任务%d由工人%d来做：%d\n", i, TempTask, Cost[TempTask]);  
}  
13、八皇后问题，输出了所有情况，不过有些结果只是旋转了90度而已。（回溯算法的典型例题，是数据结构书上算法的具体实现，大家都亲自动手写过这个程序吗？）  
#define N 8  
int Board[N][N];  
int Valid(int i, int j) {  // 判断下棋位置是否有效  
 int k = 1;  
 for(k=1; i>=k && j>=k;k++)  
  if(Board[i-k][j-k]) return 0;  
 for(k=1; i>=k;k++)  
  if(Board[i-k][j])  return 0;  
 for(k=1; i>=k && j+k<N;k++)  
  if(Board[i-k][j+k]) return 0;  
 return 1;  
}*

*void Trial(int i, int n) {  // 寻找合适下棋位置  
 if(i == n) {  
  for(int k=0; k<n; k++) {  
   for(int m=0; m<n; m++)  
    printf("%d ", Board[k][m]);  
   printf("\n");  
  }  
  printf("\n");  
 } else {  
  for(int j=0; j<n; j++) {  
   Board[j] = 1;  
   if(Valid(i,j))  
    Trial(i+1, n);  
   Board[j] = 0;  
  }  
 }  
}*

*int main(int argc, char\* argv[]) {  
 Trial(0, N);  
}*

*14、实现strstr功能，即在父串中寻找子串首次出现的位置。（笔试中常让面试者实现标准库中的一些函数）  
char \* strstring(char \*ParentString, char \*SubString) {  
 char \*pSubString, \*pPareString;  
 for(char \*pTmp=ParentString; \*pTmp; pTmp++) {  
  pSubString = SubString;  
  pPareString = pTmp;   
  while(\*pSubString == \*pPareString && \*pSubString != '\0') {  
   pSubString++;  
   pPareString++;  
  }  
  if(\*pSubString == '\0')  return pTmp;  
 }  
 return NULL;  
}*

*int main(int argc, char\* argv[]) {  
 char \*ParentString = "happy birthday to you!";  
 char \*SubString = "birthday";  
 printf("%s",strstring(ParentString, SubString));  
}*

*15、现在小明一家过一座桥，过桥的时候是黑夜，所以必须有灯。现在小明过桥要1分，小明的弟弟要3分，小明的爸爸要6分，小明的妈妈要8分，小明的爷爷要12分。每次此桥最多可过两人，而过桥的速度依过桥最慢者而定，而且灯在点燃后30分就会熄灭。问小明一家如何过桥时间最短？（原本是个小小智力题，据说是外企的面试题，在这里用程序来求解）  
#include "stdafx.h"  
#define N    5  
#define SIZE 64*

*// 将人员编号：小明-0，弟弟-1，爸爸-2，妈妈-3，爷爷-4  
// 每个人的当前位置：0--在桥左边， 1--在桥右边  
int Position[N];      
// 过桥临时方案的数组下标； 临时方案； 最小时间方案；   
int Index, TmpScheme[SIZE], Scheme[SIZE];     
// 最小过桥时间总和，初始值100；每个人过桥所需要的时间  
int MinTime=100, Time[N]={1, 3, 6, 8, 12};    
// 寻找最佳过桥方案。Remnant:未过桥人数; CurTime:当前已用时间;   
// Direction:过桥方向,1--向右,0--向左  
void Find(int Remnant, int CurTime, int Direction) {  
    if(Remnant == 0) {                               // 所有人已经过桥，更新最少时间及方案  
        MinTime=CurTime;  
        for(int i=0; i<SIZE && TmpScheme>=0; i++)  
            Scheme = TmpScheme;  
    } else if(Direction == 1) {                        // 过桥方向向右，从桥左侧选出两人过桥  
        for(int i=0; i<N; i++)                      
            if(Position == 0 && CurTime + Time < MinTime) {   
                TmpScheme[Index++] = i;  
                Position = 1;  
                for(int j=0; j<N; j++) {  
                    int TmpMax = (Time > Time[j] ? Time : Time[j]);  
                    if(Position[j] == 0 && CurTime + TmpMax < MinTime) {  
                        TmpScheme[Index++] = j;      
                        Position[j] = 1;          
                        Find(Remnant - 2, CurTime + TmpMax, !Direction);   
                        Position[j] = 0;          
                        TmpScheme[--Index] = -1;  
                    }  
                }  
                Position = 0;  
                TmpScheme[--Index] = -1;  
            }  
    } else {        // 过桥方向向左，从桥右侧选出一个人回来送灯  
        for(int j=0; j<N; j++) {  
            if(Position[j] == 1 && CurTime+Time[j] < MinTime) {  
                TmpScheme[Index++] = j;  
                Position[j] = 0;  
                Find(Remnant+1, CurTime+Time[j], !Direction);  
                Position[j] = 1;  
                TmpScheme[--Index] = -1;  
            }  
        }  
    }  
}  
int main(int argc, char\* argv[]) {  
    for(int i=0; i<SIZE; i++)   // 初始方案内容为负值，避免和人员标号冲突  
        Scheme = TmpScheme = -1;*

*Find(N, 0, 1);        // 查找最佳方案*

*printf("MinTime=%d:", MinTime); // 输出最佳方案  
    for(int i=0; i<SIZE && Scheme>=0; i+=3)  
        printf("  %d-%d  %d", Scheme, Scheme[i+1], Scheme[i+2]);  
    printf("\b\b  ");  
}*

*16、2005年11月金山笔试题。编码完成下面的处理函数。函数将字符串中的字符'\*'移到串的前部分，前面的非'\*'字符后移，但不能改变非'\*'字符的先后顺序，函数返回串中字符'\*'的数量。如原始串为：ab\*\*cd\*\*e\*12，处理后为\*\*\*\*\*abcde12，函数并返回值为5。（要求使用尽量少的时间和辅助空间）  
int change(char \*str) {     /\* 这个算法并不高效，从后向前搜索效率要高些 \*/  
 int count = 0;     /\* 记录串中字符'\*'的个数 \*/  
 for(int i=0, j=0; str; i++) {  /\* 重串首开始遍历 \*/  
  if(str=='\*') {    /\* 遇到字符'\*' \*/  
   for(j=i-1; str[j]!='\*'&&j>=0; j--) /\* 采用类似插入排序的思想，将\*前面 \*/  
    str[j+1]=str[j];     /\* 的非\*字符逐个后移，直到遇到\*字符 \*/  
   str[j+1] = '\*';  
   count++;  
  }  
 }  
 return count;  
}  
int main(int argc, char\* argv[]) {  
 char str[] = "ab\*\*cd\*\*e\*12";  
 printf("str1=%s\n", str);  
 printf("str2=%s, count=%d", str, change(str));  
}  
// 终于得到一个比较高效的算法，一个网友提供，估计应该和金山面试官的想法一致。算法如下：  
int change(char \*str) {  
 int i,j=strlen(str)-1;  
 for(i=j; j>=0; j--) {  
  if(str!='\*') {  
   i--;  
  } else if(str[j]!='\*') {  
   str = str[j];  
   str[j] = '\*';  
   i--;  
  }  
 }  
 return i+1;  
}*

*17、2005年11月15日华为软件研发笔试题。实现一单链表的逆转。  
#include "stdafx.h"  
typedef char eleType;  // 定义链表中的数据类型  
typedef struct listnode  { // 定义单链表结构  
 eleType data;  
 struct listnode \*next;  
}node;*

*node \*create(int n) {  // 创建单链表，n为节点个数  
 node \*p = (node \*)malloc(sizeof(node));   
 node \*head = p;  head->data = 'A';  
 for(int i='B'; i<'A'+n; i++) {      
  p = (p->next = (node \*)malloc(sizeof(node)));  
  p->data = i;  
  p->next = NULL;    
 }  
 return head;  
}*

*void print(node \*head) { // 按链表顺序输出链表中元素  
 for(; head; head = head->next)  
  printf("%c ", head->data);   
 printf("\n");  
}*

*node \*reverse(node \*head, node \*pre) { // 逆转单链表函数。这是笔试时需要写的最主要函数  
 node \*p=head->next;  
 head->next = pre;  
 if(p) return reverse(p, head);  
 else  return head;  
}*

*int main(int argc, char\* argv[]) {  
 node \*head = create(6);  
 print(head);  
 head = reverse(head, NULL);  
 print(head);  
}*

*18、编码实现字符串转整型的函数（实现函数atoi的功能），据说是神州数码笔试题。如将字符串 ”+123”◊123, ”-0123”◊-123, “123CS45”◊123, “123.45CS”◊123, “CS123.45”◊0  
#include "stdafx.h"  
int str2int(const char \*str) {    // 字符串转整型函数  
 int i=0, sign=1, value = 0;  
 if(str==NULL)  return NULL;    // 空串直接返回 NULL  
 if(str[0]=='-' || str[0]=='+') {   // 判断是否存在符号位  
  i = 1;  
  sign = (str[0]=='-' ? -1 : 1);  
 }  
 for(; str>='0' && str<='9'; i++) // 如果是数字，则继续转换  
  value = value \* 10 + (str - '0');  
 return sign \* value;  
}*

*int main(int argc, char \*argv[]) {  
 char \*str = "-123.45CS67";   
 int  val  = str2int(str);  
 printf("str=%s\tval=%d\n", str, val);  
}*

*19、歌德巴赫猜想。任何一个偶数都可以分解为两个素数之和。（其实这是个C二级考试的模拟试题）  
#include "stdafx.h"  
#include "math.h"  
int main(int argc, char\* argv[]) {  
 int Even=78, Prime1, Prime2, Tmp1, Tmp2;  
 for(Prime1=3; Prime1<=Even/2; Prime1+=2) {  
  for(Tmp1=2,Tmp2=sqrt(float(Prime1)); Tmp1<=Tmp2 && Prime1%Tmp1 != 0; Tmp1++);  
  if(Tmp1<=Tmp2) continue;  
  Prime2 = Even-Prime1;  
  for(Tmp1=2,Tmp2=sqrt(float(Prime2)); Tmp1<=Tmp2 && Prime2%Tmp1 != 0; Tmp1++);  
  if(Tmp1<=Tmp2) continue;  
  printf("%d=%d+%d\n", Even, Prime1, Prime2);  
 }  
}*

*20、快速排序（东软喜欢考类似的算法填空题，又如堆排序的算法等）  
#include "stdafx.h"  
#define N 10  
int part(int list[], int low, int high) {  // 一趟排序，返回分割点位置  
 int tmp = list[low];  
 while(low<high) {  
  while(low<high && list[high]>=tmp) --high;  
  list[low] = list[high];  
  while(low<high && list[low]<=tmp)  ++low;  
  list[high] = list[low];  
 }  
 list[low] = tmp;  
 return low;  
}  
void QSort(int list[], int low, int high) { // 应用递归进行快速排序  
 if(low<high) {  
  int mid = part(list, low, high);  
  QSort(list, low, mid-1);  
  QSort(list, mid+1, high);  
 }  
}  
void show(int list[], int n) {    // 输出列表中元素  
 for(int i=0; i<n; i++)  
  printf("%d ", list);  
 printf("\n");  
}  
int main(int argc, char\* argv[]) {  
 int list[N] = {23, 65, 26, 1, 6, 89, 3, 12, 33, 8};  
 show(list, N);      // 输出排序前序列  
 QSort(list, 0, N-1);     // 快速排序  
 show(list, N);      // 输出排序后序列  
}*

*21、2005年11月23日慧通笔试题：写一函数判断某个整数是否为回文数，如12321为回文数。可以用判断入栈和出栈是否相同来实现（略微复杂些），这里是将整数逆序后形成另一整数，判断两个整数是否相等来实现的。  
#include "stdafx.h"  
int IsEchoNum(int num) {  
 int tmp = 0;  
 for(int n = num; n; n/=10)  
  tmp = tmp \*10 + n%10;  
 return tmp==num;  
}*

*int main(int argc, char\* argv[]) {  
 int num = 12321;  
 printf("%d  %d\n", num, IsEchoNum(num));  
}*

*22、删除字符串中的数字并压缩字符串（神州数码以前笔试题），如字符串”abc123de4fg56”处理后变为”abcdefg”。注意空间和效率。（下面的算法只需要一次遍历，不需要开辟新空间，时间复杂度为O(N)）  
#include "stdafx.h"  
void delNum(char \*str) {  
 int i, j=0;  
// 找到串中第一个数字的位子  
 for(i=j=0; str && (str<'0' || str>'9'); j=++i);  
   
 // 从串中第一个数字的位置开始，逐个放入后面的非数字字符  
 for(; str; i++)     
  if(str<'0' || str>'9')   
   str[j++] = str;  
 str[j] = '\0';  
}*

*int main(int argc, char\* argv[]) {  
 char str[] = "abc123ef4g4h5";  
 printf("%s\n", str);  
 delNum(str);  
 printf("%s\n", str);  
}*

*23、求两个串中的第一个最长子串（神州数码以前试题）。如"abractyeyt","dgdsaeactyey"的最大子串为"actyet"。  
#include "stdafx.h"  
char \*MaxSubString(char \*str1, char \*str2) {  
 int i, j, k, index, max=0;  
 for(i=0; str1; i++)  
  for(j=0; str2[j]; j++) {  
   for(k=0; str1[i+k]==str2[j+k] && (str2[i+k] || str1[i+k]); k++);  
   if(k>max) {  // 出现大于当前子串长度的子串，则替换子串位置和程度  
    index = j; max = k;  
   }  
  }  
 char \*strResult = (char \*)calloc(sizeof(char), max+1);  
 for(i=0; i<max; i++)    
  strResult = str2[index++];  
 return strResult;  
}*

*int main(int argc, char\* argv[]) {  
 char str1[] = "abractyeyt", str2[] = "dgdsaeactyey";  
 char \*strResult = MaxSubString(str1, str2);  
 printf("str1=%s\nstr2=%s\nMaxSubString=%s\n", str1, str2, strResult);  
}*

*24、不开辟用于交换数据的临时空间，如何完成字符串的逆序(在技术一轮面试中，有些面试官会这样问)  
#include "stdafx.h"  
void change(char \*str) {  
 for(int i=0,j=strlen(str)-1; i<j; i++, j--){  
  str ^= str[j] ^= str ^= str[j];  
 }  
}  
int main(int argc, char\* argv[]) {  
 char str[] = "abcdefg";  
 printf("strSource=%s\n", str);  
 change(str);  
 printf("strResult=%s\n", str);  
 return getchar();  
}*

*25、删除串中指定的字符（做此题时，千万不要开辟新空间，否则面试官可能认为你不适合做嵌入式开发）  
#include "stdafx.h"  
void delChar(char \*str, char c) {  
 int i, j=0;  
 for(i=0; str; i++)  
  if(str!=c) str[j++]=str;  
 str[j] = '\0';  
}*

*int main(int argc, char\* argv[]) {  
 char str[] = "abcdefgh"; // 注意，此处不能写成char \*str = "abcdefgh";  
 printf("%s\n", str);  
 delChar(str, 'c');  
 printf("%s\n", str);  
}*

*26、判断单链表中是否存在环（网上说的笔试题）  
#include "stdafx.h"  
typedef char eleType;    // 定义链表中的数据类型  
typedef struct listnode  {   // 定义单链表结构  
 eleType data;  
 struct listnode \*next;  
}node;*

*node \*create(int n) {    // 创建单链表，n为节点个数  
 node \*p = (node \*)malloc(sizeof(node));   
 node \*head = p;  head->data = 'A';  
 for(int i='B'; i<'A'+n; i++) {  
  p = (p->next = (node \*)malloc(sizeof(node)));  
  p->data = i;  
  p->next = NULL;  
 }  
 return head;  
}*

*void addCircle(node \*head, int n) { // 增加环，将链尾指向链中第n个节点  
 node \*q, \*p = head;  
 for(int i=1; p->next; i++) {  
  if(i==n) q = p;  
  p = p->next;  
 }  
 p->next = q;  
}*

*int isCircle(node \*head) {   // 这是笔试时需要写的最主要函数，其他函数可以不写  
 node \*p=head,\*q=head;   
 while( p->next && q->next) {   
  p = p->next;  
  if (NULL == (q=q->next->next)) return 0;  
  if (p == q) return 1;  
 }  
 return 0;   
}*

*int main(int argc, char\* argv[]) {  
 node \*head = create(12);  
 addCircle(head, 8);   // 注释掉此行，连表就没有环了  
 printf("%d\n", isCircle(head));  
}*

<http://www.cnblogs.com/hlxs/archive/2011/09/01/2161940.html>