欢迎您参加 Shanghai Suntec 的面试！

以下是算法及 C 语言的测试题目，题目总量为 30 题，总分为 100 分，其中 P1-P10 每题 2 分，P11-P30 每题为 4 分。
30 个题目都是选择题，有的是单选，有的是多选。多选题的答案必然在 2 个以上（含 2 个）。

请将答案写在答题纸上，如有问题，请与面试人员联系。

P1：（多选）
有如下定义 int a; int *b; 则下列哪些语句是正确的：
A：b=&a;
B：b=*a;
C：b=(int*)a;
D：*b=a;
P2：（单选）
有如下定义
char * const s1="string";
char const *s2="string";
则下列哪些语句是正确的：
A：s1="w";
B：*s1='w';
C：s2="w";
D：*s2='w';
P3：（多选）
有如下定义
Char *s1="string";
char s2[ ]= "string";
则下列哪些语句是正确的：
A：*s1='w';
B：s1="w";
C：*s2='w';
D：s2="w";
P4：（单选）
有如下定义
#define SQUARE(x) x*x
则 SQUARE(SQUARE(1+1))*2 的值为：
A：6
B：10
C：18
D：32
P5：（单选）
一个"指向整型数组的指针"的定义为：
A：int (*ptr)[ ]
B：int *ptr[ ]
C：int *(ptr[ ])
D：int ptr[ ]
P6：（单选）
b=2，c=1 下列哪些语句使 a 的值为 1：
A：a=b++;
B：a=++b;
C：a=++(c+1);
D：a=(b+1)++;
E：以上都不是
P7（单选）
有如下定义 char tt[256]; 那么 sizeof(tt)的值为
A：1;
B：256;
C：4;
D：0

P8（单选）
请写出以下程序的输出结果

```
main( )
{
    char text[ ]= "abcdefghijklmno";
    int i=0;
    text[12] ='\0';
    while(text[++i]!='\0')
    {
        Printf("%c", text[i++]);
    }
}
```

A："acegik"
B："bdfhjln"
C："bdfhjl"
D："acegiko"

P9 和 P10 基于下列程序：

```
//这是一个用冒泡法排序的程序，请在空格处填入适当代码
#include <stdio.h>
#include <string.h>
void sort(char s[ ], int  nNum);
void        /* Return nothing                */
main(void)    /* Process entry point(No argument)
*/
{
   char str[ ]="OLWF";
   sort(str, 4);
   printf("%s",str);
}
void        /* Return nothing                */
sort(char  s[  ],  /*  Start  address  of  string
*/
   int  nNum)   /*  Number  of  char  in  string
*/
{
   int     i, j;
   char    temp;
   for(i=0;i<nNum;i++){
     for(j=i+1;j<nNum;j++){
         if(s[i]<s[j]){
             temp=s[i];
             __[P9]___
             __[P10]__
         }
     }
   }
}
```

P9：（单选）
A：s[i]=s[j];
B：s[j]=temp;
C：s[j]=s[i];
D：Temp=s[j];
P10：（单选）
A：s[i]=s[j];
B：s[j]=temp;
C：s[j]=s[i];
D：Temp=s[j];

```
//以下问题 P11-P20 将在内存管理，堆栈处理，和森林以及二叉树
算法的 C 语言实现进行测试
#include <stdio.h>
#include <string.h>
//以下是对于常数、类型和函数指针的定义
#define MEMORY_SIZE  102400
typedef unsigned char    BYTE;
typedef void *           PTR;
typedef PTR (*NodeFunc)(void);
typedef void (*DoFunc)(int n, PTR p);
//以下是对内存管理的类型定义和处理函数定义
typedef struct_Memory *PCMemory;
typedef struct_Memory
{
    BYTE amem[MEMORY_SIZE];
    Int       nMemPoint;
}CMemory;
Void MInit (PCMemory pmem          );  //内存初始化
PTR MAlloc(PCMemory pmem, int nSize)  //内存分配函数
//以下是对堆栈处理的类型定义和处理函数定义
typedef struct_Stack *PCStack
typedef struct_ Stack
{
    PTR pContent;
    PCStack pNext;
}CStack;
PCStack SNew(void);                          //堆栈初始化
PCStack SPush(PCStack pstk,PTR p);       //堆栈进栈
PCStack SPop(PCStack pstk,PTR *pp);      //堆栈出栈
//以下是对用二叉树表示的森林处理的类型定义和处理函数定义
typedef struct_Tree *PCTree;
typedef struct_Tree
{
    PTR pContent;
    PCTree pSon;
    PCTree pBrother;
}CTree;
PCTree TNew(void);   //初始化一个森林
//向根节点或者子树节点增加儿子和兄弟
Void TAddSon(PCTree ptree, Nodefunc f);
Void TAddBrother(PCTree ptree, Nodefunc f);
//使用递归方法和堆栈方法对树进行遍历
Void TBrowse(PCTree ptree, DoFunc f, int n);
Void TBrowseStack(PCTree ptree, DoFunc f, int n);
//以下是内存实体的全局变量定义
static CMemory mem;
PCMemory pmem=&mem;
//以下是内存管理的处理函数的实现
Void MInit(PCMemory pmem)
{
  pmem->nMemPoint=0;
}//MInit
PTR MAlloc(PCMemory pmem, int nSize)
{
  int nCurMemPoint;
  nCurMemPoint=pmem->nMemPoint;
  pmem->nMemPoint+=nSize;
  return(__[P11]__);
}//MAlloc              //P11,P12
//以下是堆栈处理的处理函数的实现
PCStack SPush(PCStack pstk,PTR p)
{
  PCStack pstkNew;
  pstrNew=MAlloc(pmem,sizeof(CStack));
  pstkNew->pContent=p;
  pstkNew->pNext=pstk;
  return(__[P13]__);  //P13
}SPush
PCStack Spop(PCStack pstk,PTR*pp)
{
  if(pstk!=NULL)   //P14
   {
      *pp=pstk->pContent;
       pstk=pstk->pNext;
       return(pstk) ;
   }
   else return(NULL);
}//SPop
PCStack SNew(void){return(NULL);}  //SNew
//以下是用二叉树表示森林的处理函数的实现
PCTree TNew(void)
{
   PCTree pRoot;
   pRoot=MAlloc(pmem,sizeof(CTree));
   pRoot->pBrother=NULL;
   pRoot->pSon=NULL;
   pRoot->pContent="ROOT";
   return(pRoot);
}//TNew     //P15
Void TAddSon(PCTree ptree,NodeFunc f)
{
  PTR p;
  PCTree ptreeNew;
  p=f();
  if(p==NULL)TAddBrother(ptree,f);
  else
  {
     ptreeNew=MAlloc(pmem,sizeof(CTree));
     ptreeNew->pBrother=NULL;
     ptreeNew->pSon=NULL;
     ptreeNew->pContent=p;
     ptreeNew->pSon= ptreeNew;
     TAddSon(ptreeNew,f);
     __[P16-1]__;   //P16
  }
}//TAddSon
void TAddBrother (PCTree ptree,NodeFunc f)
{
  PTR p;
  PCTree ptreeNew;
  p=f();
  if(p==NULL)return;
  else
  {
     ptreeNew=MAlloc(pmem,sizeof(CTree));
     ptreeNew->pBrother=NULL;
     ptreeNew->pSon=NULL;
     ptreeNew->pContent=p;
     ptreeNew->pBrother= ptreeNew;
     TAddSon(ptreeNew,f);
     __[P16-2]__;    //P16
  }
}//TAddBrother
void TBrowse(PCTree ptree,DoFunc f,int n)
{
  if(ptree!=NULL)
  {
     __[P17]__;  //P17
     TBrowse (ptree->pSon,f,n+1);
     TBrowse (ptree->pBrother,f,n);
  }
}// TBrowse
void TBrowseStack(PCtree ptree,DoFunc f,int n)
{
   PCStack pstk;
   pstk =SNew();
   while(1)
   {
      __[P17]__;
      if(ptree->pSon==NULL)
      {
        //__[P18]__;  //P18
         {
            pstk=SPop(pstk,&ptree);
```

关闭

```
            n--;
            if(pstk==NULL) return;
        }
        ptree=ptree->pBrother;
    }
    else
    {
        pstk=SPush(pstk,ptree);
        n++;
        ptree=ptree->Son;
    }
  }
}//TBrowse

//以下是测试用函数和数据
static char *aaNode[22]= {
"1","2","3",NULL, "4", NULL, "5", NULL, NULL, "6",
NULL, "7", "9", NULL, "9", "10", NULL, "11",
NULL, "12", NULL, NULL,};// P19
static int nNode=0;

PTR Node(void){return(aaNode[nNode++]);}// Node

Void Out(int n,PTR p)
{
  int i;
  for(i=0;i<n;i++) printf("_");
  printf("%s\n",p);
}//Out

//以下是主函数
void main(void)
{
  PCTree ptree;
  MInit(pmem);
  Ptree=TNew();
  TAddSon(ptree,Node);
  TBrowse(ptree->pSon,Out,0);
  TBrowseStack(ptree->pSon,Out,0);//P20
}//main
```

以下是正确的输出结果

```
1
_2
__3
__4
__5
_6
_7
__8
__9
___10
___11
___12
1
_2
__3
__4
__5
_6
_7
__8
__9
___10
___11
___12
```

P11:（单选）
[P11]处应当写的语句是
A: (void*)(&(pmem->amem[nCurMemPoint]))
B: (void*)(pmem->amem[nCurMemPoint])
C: (void*)(&(pmem->amem[pmem->nMemPoint]))
D: (void*)(pmem->amem[pmem->nMemPoint])
E: 以上都不是

P12:（多选）
对于以上内存管理的类型定义和实现函数叙述正确的是：
A: 该类型定义的空间在程序的静态存储区，不能用于动态的内存分配，应当使用 malloc 等函数进行内存分配
B: 没有进行内存分配越界的检查
C: 不能用于对长整数和双精度整数进行内存分配，否则会导致内存错
D: 没有考虑计算机字长的影响，在 Sun Workstation 等以字而不是字节为单位的机器上可能会导致内存错
E: 不能用于对用 struct 定义的复杂变量进行内存分配，否则会导致内存错

P13:（单选）
[P13]处应当写的语句是
A: pstk+1
B: pstk->pNext
C: pstk
D: pstkNew
E: 以上都不是

P14:（单选）
[P14]处语句的作用是
A: 防止堆栈指针出现循环
B: 检查堆栈内容的合法性
C: 判断堆栈是否为空
D: 防止堆栈使用的内存溢出
E: 以上都不是

P15:（多选）
根据这个创建函数可以推断，这个森林
A: 是一个按照内容排序的森林
B: 只能处理字符串
C: 有一个虚拟的根节点 pRoot，真正的根节点是 pRoot->pBrother
D: 有一个虚拟节的根点 pRoot，真正的根节点是 pRoot->Son
E: 如果利用根节点的 pBrother，可以处理多个森林的情况

P16:（单选）
[P16-1]和[P16-2]处应当写的语句是
A: [1]TAddBrother(ptree,f) [2]TAddBrother(ptree,f)
B: [1]TAddBrother(ptree,f) [2](不写任何语句)
C: [1] (不写任何语句) [2] TAddBrother(ptree,f)
D: [1]TAddBrother(ptreeNew,f)
[2] TAddBrother(ptreeNew,f)
E: [1]TAddBrother(ptreeNew,f) [2] (不写任何语句)
F: [1] (不写任何语句) [2] TAddBrother(ptreeNew,f)

P17:（单选）
[P17]处语句的功能利用函数指针是对树节点的内容进行处理，这条语句应当是
A: f(n,ptree->pContent);
B: f(p,ptree);
C: f(n,&(ptree->pContent));
D: f(n,& ptree);
E: f();

P18:（单选）
[P18]处语句的作用是
A: while(ptree==NULL)
B: if(ptree==NULL)
C: while(ptree->pSon==NULL)
D: if(ptree->pSon==NULL)
E: while(ptree->pBrother==NULL)
F: if(ptree->pBrother==NULL)

P19:（多选）
树组 aaNode 中 NULL 的意义为
A: 当前节点不再有儿子
B: 当前节点不再有兄弟
C: 当前节点的内容为空
D: 当前节点既没有儿子也没有兄弟
E: 当前节点肯定有儿子
F: 当前节点肯定有兄弟

P20:（多选）
关于本程序中使用递归方法和堆栈方法进行树遍历，以下叙述正确的是
A: 就时间复杂性而言，递归方法是 O(n)，堆栈方法是 O(nLogn)
B: 就算法复杂性而言，递归方法是 O(nLogn)，堆栈方法是 O(n)

...杂性而言，递归方法和堆栈方法是一样的
...递归方法中，需要为局部变量和参数多次分配调用栈，可能导致调用栈溢出
E．堆栈方法需要进行额外的操作，而且程序复杂，所以速度较慢

以下问题 P21-P25 基于下列程序
一个相连的区域被不规则的分割成 n 个不同的小区域，每个小区域与若干其他小区域相邻接．现用 cn 种不同的颜色为该区域着色，要求每个小区域着一种颜色，相邻小区域着不相同颜色．
设小区域被顺序编号为 0,1,…,n-1．每个小区域与其他小区域的邻接关系用二维数组 bordering 表示，元素 bordering[i][j]表示 i 号小区域与 j 号小区域之间的邻接关系：
bordering[i][j]=0：j 小区域与 i 小区域不邻接
bordering[i][j]=1：j 小区域与 i 小区域相邻接
程序中，把计算结果存放于两维数组 colored 中．颜色编号为 0,1，…，cn-1．元素 colored[color][j]的含义是：
colored[color][j]=0：j 小区域不用颜色 color 着色
colored[color][j]=1：j 小区域用颜色 color 着色
函数 colorcountry(bordering，colored，n，cn)根据所给的小区域邻接关系数组 bordering，小区域个数 n，颜色数 cn，将找到的着色方案记录在数组 colored 中．函数采用试探法找解．首先从第一个小区域着第一种颜色开始顺序为各个小区域找着色方案．对某个小区域，当为他找到一种未被他的相邻小区域着色的颜色时，就用该颜色对该小区域着色，并准备处理下一个小区域．当不能为某个小区域找到一个未被他的相邻的小区域着色的颜色时，就回溯．如果最终为全部小区域找到着色方案，函数返回 1；否则，函数返回 0．
程序假定小区域个数不超过 20，着色数为 4．

```
//程序
#include"stdio.h"
#define N 20
#define CN 4
int    colorcountry(int    bordering[    ][N],int
colored[ ][N], int n,int cn)
{
   int color,used,i,c;
   for(color=0;color<cn;color++)
    /*设置所有区域未着色*/
     for(i=0;i<n;i++)
      colored[color][i]=0;
      c=0;    /*从第一个小区域开始*/
      color=0; /*从第一种颜色开始试探*/
    while(c<n)
   {/*还未对全部小区域着色时循环*/
      while(__[P21]__) /*顺序对每种颜色做试探*/
       { /*检查当前颜色是否已被某相邻小区域着色*/
          for(i=0,used=0;!used&&i<c;i++)
            if(__[P22]__)
              used=1;
            if(!used)
              break;  /*当前颜色未被某相邻小区域着色
*/
            color++
        }
        if(!used)
        {
          /*找到可用颜色，用此色着色，并准备处理下一小
区域*/
          __[P23]__;
          color=0;
        }
        else
        { /*未找到一种可用颜色，回溯*/
          c--;
          if(c<0)
            return 0;  /*发现没有解的情况*/
          for(color=0;__[P24]__;color++)
          __[P25]__;
        }
   }
   return 1;
}
Void print (int colored[ ][N],int n, int cn)  /*输
出结果*/
{
  char
*colortbl[ ]={"RED","BLUE","GREEN","YELLOW"};
  int color,i;
  for (color =0;color<cn;color++)
  {
     printf("\n%s:\n",colortbl[color]);
     for(i=0;i<n;i++)
       ff(colored[color][i])
          printf("\t%d",i);
          printf("\n");
   }
}
int colored[CN][N],bordering[N][N];
void main(void)
{
  int i,j,n;
  printf("enter number of areas.");
  scanf("%d",&n);
  printf("enter bordering:\n");
  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
      bordering[i][j]=0;
for(i=0;i<n;i++)
{
  printf("enter  areas  to  link  %d  area(<  0  to
next).\n",i);
  scanf("%d",&j);
  while(j>=0)
  {
    if(i!=j)
      bordering[i][j]=bordering[j][i]=1
      scanf("%d",&j);
   }
 }
if (colorcountry(bordering,colored,n,cn))
  print(colored,n,cn);
else
 printf("no solution.\n");
}
```

//问题（P21-P25 都是单选）
P21：
A：color<=cn
B：color<cn
C：color++<=cn
D：color++<cn
P22：
A：bordering[c++][i]==1&&colored[color][i]==1
B：bordering[c+1] [i]==1&&colored[color][i]==1
C：*(bordering[c+1]+1)==1&&colored[color][i]==1
D：bordering[c][i]==1&&colored[color][i]==1
P23：
A：colored[color][c]=1
B：colored[color][c+1]=1
C：colored[color][++c]=1
D：colored[color][c++]=1

A: colored[color][c]==0
B: colored[color][c]
C: colored[color+1][c]==0
D: colored[color+1][c]
P25
A: colored[color+1][c]=0
B: colored[color-1][c]=0
C: colored[color][c]=0
D: colored[color++][c]=0

本程序是一个简单的计数器模拟程序，对任给的正确四则运算表达式，程序计算其结果值并输出。表达式中运算分量为无正负号整数，运算符为+、-、*、/,圆括号按常规配对，表达式以字符"="结束。函数 getach()为获取表达式的一个合法字符，并将字符存入变量 curch；函数指针数组 func[ ]是为了统一加减乘除计算而设置的。

//程序

```
#include<stdio.h>
#include<string.h>
int add(int x,int y) {return x+y;}
int sub(int x,int y) {return x-y;}
int mul(int x,int y) {return x*y;}
int div(int x,int y) {return x/y;}
int (*func[])(int x, int y)={add,sub,mul,div};
int num,curch;
char chtbl[ ]="+-*/()=";
char corch[ ]="+-*/()=0123456789";
int getach()
{
   unsigned int i;
   while(true)
   {
    curch=getchar();
     if(curch==eof)return -1;
     for(i=0;corch[i]&&curch!=corch[i];i++);
     if(i<strlen(corch))break;
   }
return curch;
}
int getid()
{
 int i;
 if(curch>='0'&&curch<='9')
 {
  for(num=0;curch>='0'&&curch<='9';getach())
    num=__[p26]__;
    return -1;
 }
 else
 {
  for(i=0;chtbl[i];i++)
    if(chtbl[i]==curch)break;
   if(i<=5) getach();
 return i;
 }
}
int cal()
{
 int x1, x2, x3, op1, op2, i;
 i =getid();
 if(i==4)
  x1=cal();
 else
   x1=num;
 op1=getid();
 if(op1>=5)return x1;
 i=getid();
 if(i==4)
   x2=cal();
 else
   x2=num;
 op2=getid();
 while(__[p27]__)
 {
  i=getid();
  if(i==4)
    x3=cal();
  else
    x3=num;
  if((op1/2==0)&&(op2/2==1))
    x2=func[op2](x2,x3);
  else
  {
     x1=__[p28]__;
     x2=x3;
     __[p29]__;
  }
  op2=getid();
 }
return___[p30]___
}
void main()
{
   int value;
   printf ("please input an expression:\n");
   getach();
   while(curch!='=')
   {
     value=cal();
     printf("the result is :%d\n",value);
     printf("please input an expression:\n");
     getach();
   }
}
```

//问题（P26-P30 都是单选）

P26：
A: (num+1)*10+curch-1-'0'
B: (num+1)*10+curch-'0'
C: num*10+curch-'0'
D: num*10+curch-1-'0'

P27：
A: (op2>=0)&&(op2<6)
B: op2<6
C: (op2>=0)&&(op2<5)
D: op2>=0

P28：
A: func[op2](x1,x2)
B: (*func[op2])(x1,x2)
C: (*func[op1])(x1,x2)
D: func[op1](x1,x2)

P29：
A: op2=op1
B: op1=op2
C: x3=func[op1](x1,x2)
D: x3=(*func[op1])(x1,x2)

P30：
A: func[op2](x1,x2)
B: (*func[op2])(x1,x2)
C: (*func[op1])(x1,x2)
D: func[op1](x1,x2)