YDSP Senior 组模拟考试

2023 年 9 月 10 日

本次考试的答题时间为两个小时，请各位选手自行掌握时间。
提交请访问 yundouxueyuan.com 参加对应的比赛，将最终答案以程序的形式提交至对应题目。
更进一步的提交细节，请参考对应题目里的说明。

最后，本次模拟考试为纯公益目的，不得用于任何未经授权的盈利活动。

* 祝考试顺利 *

2023 云斗学院软件能力认证第一轮
(YDSP - Senior) 提高级 C++ 语言试题

考生注意事项:

- 试题纸一共页，满分 100 分。作答后请记录答案，并按要求提交至对应比赛处。
- 考试过程中不得使用任何电子设备或查阅任何书籍资料。

1 选择题（每题 2 分，共 30 分）

1.1 第 1 题

被誉为“人工智能之父”的人是(　)。

A. 冯·诺依曼

B. 阿兰·图灵

C. 巴贝奇

D. 弗雷德里克·特曼

1.2 第 2 题

4 在模 7 意义下的加法逆元是(　)。

A. 2

B. 3

C. 4

D. 5

1.3 第 3 题

在第二轮认证的四个传统题中，下列四个程序中恰有一个拿到分数，它是(　)。

A. 在 `centroid.cpp` 内写了 `freopen("centorid.in","r",stdin);`

B. 同一文件夹下有 `fast.h`，然后程序开头写了 `#include"fast.h"`

C. 在源代码内写出了所有 10^7 以内的质数。

D. 在区间 DP 题中使用状态压缩 DP。

1.4 第 4 题

下列排序算法中，渐进复杂度最优的稳定算法是(　)。

A. 归并排序

B. 堆排序

C. 快速排序

D. 插入排序

1.5 第 5 题

ST 表不可以用于维护(　)。

A. 区间最大值

B. 区间最大公约数

C. 区间按位或

D. 区间按位异或

1.6 第 6 题

在 C++ 中，`0110^101` 的值的十进制表示是(　)。

A. 1

B. 3

C. 11

D. 45

1.7 第 7 题

k 至少是(　)，才能保证 6 个结点 k 条边的简单有向图一定强联通。

A. 11

B. 22

C. 26

D. 30

1.8 第 8 题

在 NOI Linux 系统的终端内，如果程序进入死循环，应按下(　)终止程序。

A. `Ctrl+C`

B. `Alt+F4`

C. `Ctrl+Z`

D. `Ctrl+W`

1.9 第 9 题

现有 $4n^2$ 个结点，n^4 条有向无权边的图，保证任意两点间最短路不超过 $4n$，那么优先队列优化 Dijkstra 求结点 1 到所有结点最短路的时间复杂度是(　)。

A. $O(n^2 \log n)$

B. $O(n^3 \log n)$

C. $O(n^4)$

D. $O(n^4 \log n)$

1.10 第 10 题

一棵满二叉树的根为 1，x 的左孩子编号为 $2x$，右孩子编号为 $2x+1$。

一同学使用如下代码求 x, y 的最近公共祖先。从算法核心思想看，该代码使用的算法是(　)。

```
int lca(int x,int y){
  if(x>y)swap(x,y);
  while((x^y)>x)
    y>>=1;
  while(x!=y)
        x>>=1,y>>=1;
  return x;
}
```

A. 枚举
B. 倍增
C. 欧拉序+ RMQ
D. 树链剖分

1.11 第 11 题

如果 $T(1)=1$，且 $T(n)=4T(\lfloor\frac{n}{2}\rfloor)+n^2$，那么 $T(n)$ 是(　　)的。
A. $O(n^2)$
B. $O(n^2\log n)$
C. $O(n^2\log^2 n)$
D. $O(n^3)$

1.12 第 12 题

一棵有根树中每个非叶结点（共 1000 个）的孩子数一定是 3 或 4。已知这棵树有 2023 个叶子结点，那么有 3 个孩子的结点有(　　)个。
A. 22
B. 23
C. 977
D. 978

1.13 第 13 题

设输出为 c，则 c 在下列(　　)区间内。

```
#include<bits/stdc++.h>
using namespace std;
int cnt=0;
int main()
{
  for(int i=1;i<=100;i++)
    for(int j=1;j<i;j++)
      for(int k=1;k<j;k++)
        for(int l=1;l<=100-i;l++)
          cnt++;
```

```
  printf("%lld\n",cnt);
}
```

A. $[9 \times 10^7, 1.1 \times 10^8]$

B. $[3 \times 10^6, 5 \times 10^6]$

C. $[9 \times 10^5, 1.1 \times 10^6]$

D. $[3 \times 10^4, 5 \times 10^4]$

1.14 第 14 题

稀疏 7 阶矩阵 A 中 $A_{1,6}, A_{2,3}, A_{3,2}, A_{4,1}, A_{4,5}, A_{5,6}, A_{6,4}$ 为 1，其余元素为 0。

那么，A^{2023} 中，非零元素有(　　)个。

A. 36

B. 28

C. 20

D. 18

1.15 第 15 题

假设定义日期的哈希函数为这个日期对应的星期（星期日为 0），例如 2023/9/16 是星期六，所以哈希值为 6。现考虑将下列日期按顺序放进哈希表，如有冲突，会往后找第一个空的位置存储（到 6 冲突了则从 0 往后），现在要依次存储 2023/9/9，2021/11/20，1999/8/1，2022/10/29，那么最后一个日期存储在哈希表的哪个地址中？

A. 0

B. 1

C. 2

D. 6

2 阅读程序（无特殊说明时判断 1.5 分，选择 3 分，3 题共 40 分）

2.1 第 1 题（13 分）

```
#include <bits/stdc++.h>
using namespace std;
const int N = 200000 + 10;
int n;
long long a[N];
struct node{
  long long md, id;
} res[4][N];
bool cmp(node a, node b){
  return a.md > b.md;
}
long long ab(long long x){
```

```
13    return x > 0 ? x : -x;
14  }
15  bool check3(long long a, long long b, long long c){
16    if(a * b * c == 0) return 0;
17    return a >= 2 && b >= 2 && c >= 2 && ab(a - b) > 1 && ab(b - c) > 1 && ab(c -
18  }
19  bool check2(long long a, long long b){
20    if(a * b == 0) return 0;
21    return b >= 3 && a >= 2 && (b - a > 2 || a - b >= 2);
22  }
23  int main(){
24    cin >> n;
25    for(int i = 1 ; i <= n ; i++)
26      cin >> a[i];
27    long long ans = 0, sum = 0;
28    for(int i = 1 ; i <= n ; i++){
29      res[1][i].md = a[i - 1] % a[i];
30      res[2][i].md = res[1][i - 1].md % a[i];
31      res[3][i].md = res[2][i - 1].md % a[i];
32      res[1][i].id = res[2][i].id = res[3][i].id = i;
33      sum += a[i];
34    }
35    for(int i = 1 ; i <= n ; i++){
36      res[1][i].md = a[i - 1] + a[i] - res[1][i].md;
37      if(i > 1)
38        res[2][i].md = a[i - 2] + a[i - 1] + a[i] - res[2][i].md;
39      if(i > 2)
40        res[3][i].md = a[i - 3] + a[i - 2] + a[i - 1] + a[i] - res[3][i].md;
41    }
42    sort(res[1] + 1, res[1] + n + 1, cmp);
43    sort(res[2] + 1, res[2] + n + 1, cmp);
44    sort(res[3] + 1, res[3] + n + 1, cmp);
45    for(int i = 1 ; i <= min(n, 8) ; i++)
46      for(int j = 1 ; j <= min(n, 8) ; j++)
47        for(int k = 1 ; k <= min(n, 8) ; k++)
48          if(check3(res[1][i].id, res[1][j].id, res[1][k].id))
49            ans = max(ans, res[1][i].md + res[1][j].md + res[1][k].md);
50    for(int i = 1 ; i <= min(n, 8) ; i++)
51      for(int j = 1 ; j <= min(n, 8) ; j++)
52        if(check2(res[1][i].id, res[2][j].id))
53          ans = max(ans, res[1][i].md + res[2][j].md);
54    ans = max(ans, res[3][1].md);
```

```
55    cout << sum - ans;
56    return 0;
57  }
```

输入数据满足 $1 \le n \le 2 \times 10^5$，$a_i \le 10^9$。

2.1.1 判断题

1. 删去第 17 行的 `a >= 2 && b >= 2 && c >= 2 &&` 对输出不造成影响。

2. 可以将 `a[0]` 设为 `INT_MAX - 1`。

3. （1 分）$a_i = 0$ 会导致程序出现运行错误。

2.1.2 选择题

4. 输入 `4 7 2023 4 2` 时，则输出为 __▲__

A. `0`

B. `1`

C. `2`

D. `3`

分别记第 45～49 行、第 50～53 行、第 54 行为【块 1】、【块 2】、【块 3】，据此完成第 5、6 两题。

5. 当输入为 `6 10 20 15 20 15 20` 时，最佳答案在 __▲__ 处取得。

A. 【块 1】

B. 【块 2】

C. 【块 3】

D. 【块 2】与【块 3】

6. 实际上，【块 1】与【块 2】的实现依赖的是一种避免分类讨论的枚举做法。我们考虑优化【块 2】。注意到【块 2】中 i, j 的上界均为 8（仅考虑 $n \ge 8$ 的情况），实际上，当 i 的上界为 __▲__，j 的上界为 __▲__ 时，便可以保证取到【块 2】的最优答案。

A. $5, 5$

B. $4, 5$

C. $5, 4$

D. $4, 4$

2.2 第 2 题（14 分）

```cpp
#include <bits/stdc++.h>
using namespace std;
int n;
long long a[300005];
long long tar;
long long pre[300005], suf[300005];
int main()
{
    cin >> n;
    cin >> tar;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    for (int i = 1; i <= n; i++) {
        pre[i] = pre[i - 1] + a[i];
    }
    if (tar <= 2LL * pre[n]) {
        for (int i = 1; i <= n; i++) {
            a[i + n] = a[i + n + n] = a[i];
        }
        n *= 3;
        for (int i = 1; i <= n; i++) {
            pre[i] = pre[i - 1] + a[i];
        }
        for (int l = 1, r = 0; l <= n; l++) {
            while (r < n && pre[r] - pre[l - 1] < tar) {
                r++;
            }
            if (pre[r] - pre[l - 1] == tar) {
                cout << "Yes\n";
                return 0;
            }
        }
        cout << "No\n";
        return 0;
    }

    for (int i = n; i >= 1; i--) {
        suf[i] = suf[i + 1] + a[i];
    }
    for (int l = 1, r = 0; l <= n + 1; l++) {
```

```
42          long long ok = (tar - suf[l]) % pre[n];
43          if (pre[r] < ok) {
44              while (r < n && pre[r] < ok) {
45                  r++;
46              }
47              if (pre[r] == ok) {
48                  cout << "Yes\n";
49                  return 0;
50              }
51          } else {
52              while (r >= 1 && pre[r] > ok) {
53                  r--;
54              }
55              if (pre[r] == ok) {
56                  cout << "Yes\n";
57                  return 0;
58              }
59          }
60      }
61      cout << "No\n";
62      return 0;
63  }
```

输入数据保证：

$1 \le n \le 10^5, 1 \le a[i] \le 10^9, 1 \le tar \le 10^{18}$

2.2.1 判断题

1. （1 分）将程序第 4 行的 `long long` 改成 `int`，程序运行依然正确。
2. （1 分）将程序第 22 行的 `i=1` 改成 `i=n/3`，程序运行依然正确
3. （1 分）将程序第 42 行的括号去掉，程序运行依然正确。
4. （2 分）每次执行第 26 行的 `while` 循环时，循环次数最大为 $O(n)$ 次。

2.2.2 选择题

5. （2 分）运行以下三组数据，得到的结果分别为？

```
7 10
7 2 4 5 1 9 2

7 100
8 5 3 5 6 2 4

```

```
6 100
1 1 4 5 1 4
```

A. Yes Yes No

B. Yes No Yes

C. Yes No No

D. No No Yes

6. 该程序的时间复杂度是 ▲ ？

A. $O(n)$

B. $O(n \log n)$

C. $O(n \log^2 n)$

D. $O(n^2)$

7. 该程序的功能是 ▲ ？

A. 判断序列 a 是否存在一个区间，其和为 tar。

B. 将 a 复制三遍得到 b，判断序列 b 是否存在一个区间，其和为 tar。

C. 将 a 复制无数遍得到 b，判断序列 b 是否存在一个区间，其和为 tar。

D. 将 a 复制无数遍得到 b，判断序列 b 是否存在一个区间，其和模 $pre[n]$ 为 tar。

2.3 第 3 题（13 分）

```cpp
#include<bits/stdc++.h>
using namespace std;
unsigned long long s,f[20][105];
int m,len;
unsigned long long dval=1;
int main(){
  cin>>s>>m;
  for(int i=0;i<10;i++)
    f[0][i%m]++;
  long long tmp=s;
  for(int k=1;tmp>9;k++,tmp/=10){
    len++; dval*=10;
    for(int r=0;r<m;r++)
      for(int j=0;j<10;j++)
        f[k][(r+j)%m]+=f[k-1][r];
  }
  int pres=0;long long sum=0;
```

```
18    for(;len>0;len--){
19      int d=s/dval%10;
20      for(int i=0;i<d;i++)
21        sum+=f[len-1][(2*m-pres-i)%m];
22      pres=(pres+d)%m;
23      dval/=10;
24    }
25    for(int i=0;i<=s%10;i++)
26      if((pres+i)%m==0)
27        sum++;
28    cout<<sum;
29    return 0;
30  }
```

输入保证 $1 \le s \le 10^{18}$，$1 \le m \le 100$。

2.3.1 判断题

1. 本程序会输出一个正整数。（2 分）
2. 第 19 行改成 `int d = s % (10 * dval) / dval`，结果不变。（1 分）
3. 其他代码不变，第 11 ~ 16 行中 `f` 可以使用滚动数组存储。（1 分）

2.3.2 选择题

1. 输入 `2023 4`，输出是 __▲__。

A. 503

B. 504

C. 505

D. 506

2. 输入 `1001001000 9`，输出是 __▲__。

A. 111222333

B. 111222334

C. 111222335

D. 111222336

3. 如果第 22 行改成了 `pres += d`，那么第 21 行可以改成 sum+ = f[len − 1][__▲__]。

A. `(m << 20 - pres - i) % m`

B. `m - (pres + i) % m`

C. `pres + i ? m - 1 - (pres + i - 1) % m : 0`

D. `pres % m + i >= m ? pres % m + i - m : pres % m + i`

3 完善程序（共两道题 30 分）

3.1 问号变换（15 分，每空 3 分）

给出一个由小写字母组成的字符串 s，里面除了小写字母，最多还包含了一个 ? 字符。其中 ? 可以表示空字符或者任意一个小写字母，求出最多有多少个子串可能使得这个子串中所有出现过的字母的个数均为偶数个。

测试数据满足 $1 \leq |s| \leq 210^4$。

例如，a?b 中，有 a?、?b、? 符合要求，其中 ? 分别表示 a、b、空格。

试补全程序。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 20010;
char s[N];
int p[N];
long long count(int len) {
  return ((len * (len - 1)) >> 1);
}
long long pure(int n) {
  long long ans = 0, x = 0;
  p[0] = 0;
  for (int i = 1 ; i <= n ; i++) {
    if (s[i] != -1) x ^= (1 << s[i]);
    p[i] = x;
  }
  sort(p, p + n + 1);
  for (int i = 0 ; i <= n ; ) {
    int t = p[i], cnt = 0;
    while (p[i] == t) i++, cnt++;
    ans += /* Blank1 */ ;
  }
  return ans;
}
int main() {
  long long ans = 0;
  scanf("%s", s + 1);
  int n = strlen(s + 1), idx = -1;
  for (int i = 1 ; i <= n ; i++) {
    if (s[i] == '?') idx = i;
      /* Blank2 */ ;
  }
  if (idx == -1) {
    printf("%lld\n", pure(n));
```

```
34      return 0;
35    }
36    for (int i = 1 ; i <= 27 ; i++) {
37      s[idx] =    /* Blank 3 */  ;
38      ans += pure(n);
39    }
40    long long cf = pure(idx - 1), len = /* Blank 4 */ ;
41    for (int i = 1 ; i <= len ; i++) /* Blank 5 */  ;
42    cf += pure(len);
43    printf("%lld\n", ans - 26 * cf);
44    return 0;
45  }
```

1. 第 /* Blank 1 */ 空应填 __▲__

A. `count(cnt + 1)`

B. `count(cnt)`

C. `count(i + 1)`

D. `count(i)`

2. 第 /* Blank 2 */ 空应填 __▲__

A. `s[i] = s[i] - 'a' - 1`

B. `s[i] = 1 << (s[i] - 'a' - 1)`

C. `s[i] = s[i] - 'a'`

D. `s[i] = 1 << (s[i] - 'a')`

3. 第 /* Blank 3 */ 空应填 __▲__

A. `i`

B. `i - 1`

C. `(i > 1) ? 0 : i`

D. `i - 2`

4. 第 /* Blank 4 */ 空应填 __▲__

A. `n - 1`

B. `n - idx`

C. `n - idx + 1`

D. `n - idx - 1`

5. 第 /* Blank 5 */ 空应填 __▲__

A. `s[i] = s[i + (i > idx)]`

B. `s[i] = s[i + (i >= idx)]`

C. `s[i] = s[i + idx]`

D. `s[i] = s[i + idx - 1]`

3.2 表达式求值（共 15 分，每空 2.5 分）

3.2.1 知识背景

为了解决以下问题，我们现在需要引入一个数据结构：语义树。当然，我们即将处理的语义仅

限于表达式中的常见运算符，因此称之为表达式树更为准确。

表达式树通常用于表示数学表达式，其中树的节点表示操作符（如加法、乘法）和操作数（如数字、变量）。这种树结构用于存储和计算表达式的值，因此它主要关注表达式的计算。

例如，对于数学表达式 `2 + 3 * 4`，相应的表达式树可以是：

```
    +
   / \
  2   *
     / \
    3   4
```

表达式树通常是一个递归结构，其中每个节点都可以有子节点。因此我们往往选择递归地构建表达式树，从根节点开始，逐步处理子表达式，直至到叶子节点、完成建树。而在通过表达式树求值时，最自然的方式便是通过*左-右-根*的后序遍历逻辑来完成计算。

3.3 问题描述

表达式求值是一个经典问题。但现在我们要对这个问题进行扩展，要求可以处理的运算符包括：`+`，`-`，`*`，`/`，`%` /*整数取余*/，`^` /*乘方*/，`(` /*左括号*/，`)` /*右括号*/ 。

同时，我们还需要处理两种特殊情况：

- 如果判断出表达式有错误，或者违背了一般意义下的整数运算规则，则输出 `error.`。
- 负数有时会不加括号。在表达式中，如果操作数出现负数（例如-8），则要特别注意。例如：10 加 -8 表示为：`10+-8`，10 减 -8 表示为：`10--8`。数据保证不会出现 10–(-8) 的情况。

值得注意的是，其中乘方运算和其他运算不同，其结合性为右结合。即对于式子 `2^3^2`，其结果应该为 `2^9 = 512`。

现在给定一个长度 $\leq 10^6$ 的满足上述要求的表达式，请输出它的结果或者 `error.`。本题采用多测，数据组数 $T \leq 20$。

请根据上述题目描述，完成接下来的代码填空问题。

```cpp
#include <bits/stdc++.h>

const int MAXN = 1000010 ;

using namespace std ;

int tp ;
int root ;
int N, T ;
bool errmsg ;

char In[MAXN] ;
char op[MAXN] ;

int stk[MAXN] ;
```

```
int nxt[MAXN] ;
int brac[MAXN] ;
int ans, val[MAXN] ;
int L[MAXN], R[MAXN] ;

bool isop(const char &x){
  return x == '+' || x == '-' || x == '*' || x == '/' || x == '%' || x == '^' ;
}
int build(const int &l, const int &r){
  if (brac[l] == r)
      return /*Blank 1*/ ;
  int rt = 0, ls = 0, rs = 0, x = 0 ;
  for (int k = l ; k <= r ; ++ k){
      if (In[k] == '(')
        k = brac[k] + 1 ;
      if (k >= r) break ;
      if ((In[k] == '+' || In[k] == '-') && k != l && !isop(In[k - 1]))
        rt = k ;
      if (In[k] == '*' || In[k] == '/' || In[k] == '%')
      ls = k ;
      if (nxt[rs] != '*' && In[k] == '^' && !rs)
      /*Blank 2*/ ;
  }
  if (rt) x = rt ;
  else if (ls) x = ls ;
  else x = rs ;
  if (!x) {
      x = r, op[x] = '?' ;
    /* Blank 3*/  ;
      return x ;
  }
  op[x] = In[x] ;
  L[x] = build(l, x - 1) ;
  R[x] = build(x + 1, r) ;
  return x ;
}
int calc(int x){
  if (op[x] == '?')
      return val[x] ;
  int res = 0, l, r ;
  l = calc(L[x]) ;
  r = calc(R[x]) ;
```

```
58    if (op[x] == '+') res = l + r ;
59    if (op[x] == '-') res = l - r ;
60    if (op[x] == '*') res = l * r ;
61    if (op[x] == '/'){
62        if (!r){
63          errmsg = 1 ;
64          return 1 ;
65        }
66        res = l / r ;
67    }
68    if (op[x] == '%') {
69        if (/* Blank 4*/){
70          errmsg = 1 ;
71          return 1 ;
72        }
73        res = l % r ;
74    }
75    if (op[x] == '^') {
76        if (r < 0){
77          errmsg = 1 ;
78          return 1 ;
79        }
80        res = pow(l, r) ;
81    }
82    return res ;
83  }
84  int main(){
85    cin >> T ;
86    while (T --){
87
88        int lst = 0 ;
89        errmsg = tp = 0 ;
90        fill(nxt + 1, nxt + N + 1, 0) ;
91
92        cin >> In + 1 ;
93        N = strlen(In + 1) ;
94
95        for (int i = 1 ; i <= N ; ++ i){
96          char lt = In[i - 1] ;
97          if (isop(In[i])){
98          nxt[lst] = i, lst = i ;
99            if (isop(lt)){
```

```
100             if (In[i] != '-')
101               goto ERR ;
102             else if (/* Blank 5*/)
103               goto ERR ;
104           }
105         }
106           else if (In[i] == '('){
107             stk[++ tp] = i ;
108             if (lt == ')' || isdigit(lt))
109             goto ERR ;
110           }
111           else if (In[i] == ')'){
112             if (!tp)
113             goto ERR ;
114             if (lt == '(')
115             goto ERR ;
116             if (isdigit(In[i + 1]))
117             goto ERR ;
118             brac[i] = stk[tp] ;
119           brac[stk[tp]] = i ;
120           tp -- ;
121           }
122         }
123         if (/* Blank 6*/) {
124           goto ERR ;
125         }
126         root = build(1, N) ;
127         ans = calc(root) ;
128
129       fill(In + 1, In + N + 1, '\n') ;
130
131         if (errmsg)
132       ERR : puts("error.") ;
133         else  printf("%d\n", ans) ;
134     }
135     return 0 ;
136 }
```

1. 第 /* Blank 1 */ 空应该填 __▲__

A. `build(l + 1, r - 1)`

B. `build(l + 1, r)`

C. `build(l, r - 1)`

D. `build(l - 1, r + 1)`

2. 第 /* Blank 2 */ 空应该填 __▲__

A. `ls = rs`

B. `ls = k`

C. `rs = k`

D. `rs = ls`

3. 第 /* Blank 3*/ 空应该填 __▲__

A. `sscanf(In + l + 1, "%d", &val[x]) ;`

B. `sscanf(In + l, "%d", &rt) ;`

C. `sscanf(In + l + 1, "%d", &rt) ;`

D. `sscanf(In + l, "%d", &val[x]) ;`

4. 第 /* Blank 4 */ 空应该填 __▲__

A. `r >= 0`

B. `r <= 0`

C. `r >= 1`

D. `r <= 1`

5. 第 /* Blank 5 */ 空应该填 __▲__

A. `!isdigit(In[i + 1])`

B. `!isop(In[i + 1])`

C. `!isdigit(In[i - 1])`

D. `In[i] == '-'`

6. 第 /* Blank 6 */ 空应该填 __▲__

A. `lst != 0`

B. `lst == 0`

C. `tp > 0`

D. `tp < 0`