工号后 6 位 (请只填工号, 不要填其他信息):_____

考试时长:3 小时, 总分:100 分

试题背景说明：

(1) 基本数据类型定义：

```
typedef  char *              LPSTR;
typedef  unsigned   char **  LPLPSTR;
typedef  signed     char     CHAR;
typedef  unsigned   char     BOOL8;
typedef  unsigned   char     BYTE;
typedef  unsigned   short    WORD;
typedef  unsigned   long     DWORD;
typedef  unsigned   short    WORD16;
typedef  unsigned   long     WORD32;
```

(2) 数据接口调用函数原型说明：

```
void dbCall( WORD wEvent, LPSTR pReq, LPSTR ptAck );
```

其中:wEvent表示事件号,pReq表示入参结构体指针,ptAck表示出参结构体指针。入参结构体的定义一般如下：

```
typedef struct
{
    /* 消息类型同步调用或异步调用,调用者必须填写该参数: */
    BYTE      bMsgType;
    ......
}D_XXX_REQ, * LPD_XXXX_REQ;
typedef struct
{
    /* 接口调用结果—返回成功或者失败码,    */
    /* 接口必须返回该参数的,供调用者使用。 */
    WORD     wRetCode;
    ......
}D_XXX_ACK,* LPD_XXX_ACK;
```

1 改错题, 每道题目至少 1 处错误或者隐患, 需要在错误的位置指明原因, 并对其进行修正 (共 60 分)。

1.1 请找出下面代码中的隐患或者错误, 说明故障原因并改正。(6 分)

```
T_REGRET regGetCgCfg(TENUM_REG_INST_TYPE tInstType, LPVOID pData,
   BYTE bDir)
{
    DM_GETCGCFG_REQ tGetCGCfgReq = {0};
    DM_GETCGCFG_ACK tGetCGCfgAck;
    T_REGRET        tRegRet = {0};
    tRegRet.retCtrl = REG_ERROR;
    if ( NULL == pData )
    {
        return tRegRet;
    }

    if (REG_INST_UEREG == tInstType)
    {
        tGetCGCfgReq.wStrategy = ((T_REG_UEINST *)pData)->wCGID;
        tGetCGCfgReq.wType = SIP_REGISTER;
    }
    else if(REG_INST_THIRD == tInstType)
    {
        tGetCGCfgReq.wStrategy =((T_REG_UEINST *)pData)->wCGDestCode;
        tGetCGCfgReq.wType = SIP_REGISTER;
    }

    tGetCGCfgReq.msgType = MSG_CALL;
    tGetCGCfgReq.bDirection = bDir;

    dbCall(DM_GETCGCFG  ,(LPSTR)&tGetCGCfgReq,(LPSTR)&tGetCGCfgAck);
    if (tGetCGCfgAck.retCODE != RC_OK)
    {
        tRegRet.retServ = REG_NOSERV;
        return tRegRet;
    }

    switch (tGetCGCfgAck.wCGPolicy)
    {
        case DB_CALLGAPING_NEED:    /* 需要查询 GDM */
            tRegRet.retCtrl = REG_SUCCESS;
            return tRegRet;
        default:    /* DB 返回异常不影响正常流程*/
            tRegRet.retCtrl=REG_SUCCESS;
            tRegRet.retServ= 0;
            return tRegRet;
    }
}

T_REGRET regThirdSaveRtaRsp(T_REG_THIRDINST *pData, LPSTR pMsgPara)
{
    T_REGRET tRegRet = {0};

    if (pData == NULL || pMsgPara == NULL)
    {
        tRegRet.retCtrl=REG_ERROR;
        return tRegRet;
    }
```

```
    ...... //省略
    if(MMCmd_SysPerfGlobal_IsEnable(MML_PERF_GLOBAL_CODE_CALLGAPING))
    {
        tRegRet=regGetCgCfg(REG_INST_THIRD,(LPVOID)pData,
            B_CALLDIR_OUT);
        if (tRegRet.retCtrl != REG_SUCCESS)
        {
            if (tRegRet.retServ = REG_NOSERV)
            {
                tRegRet.retCtrl = REG_SUCCESS;
                tRegRet.retServ = 0;
                return tRegRet;
            }
            else
            {
                tRegRet.retServ = REGUE_SERVERERROR;
                return tRegRet;
            }
        }
        else
        {
            ...... //省略
        }
    }
    return tRegRet;
}
```

1.2 请找出下面代码中的隐患或者错误,说明故障原因并改正。(7 分)

```
/*****************************************************************
* 函数名称: DataSearchByKey
* 功能描述: 根据应用传入的关键字查找实例数据区
* 输入参数: T_UNIV_HASH_KEY *pkey 输入数据的  key 值
* 输出参数: WORD32 * pdwInst 返回实例的数组的索引
*           LPVOID * ppAreaAddr 返回实例的数据区地址
* 返回值  : 正确返回 S_OK, 其他返回则异常
*****************************************************************/
T_RESULT DataSearchByKey (T_UNIV_HASH_KEY *pkey, WORD32 * pdwInst,
                          LPVOID * ppAreaAddr);

/*****************************************************************
* 函数名称: DataUnivIterNext
* 功能描述: 根据应用传入的索引查找该记录的下一个记录的索引
* 函数参数: WORD32 * pdwInst 输入参数作为找该索引对应记录的下一个纪录
* 输出参数,将下一个记录的索引带出
*           LPVOID * ppAreaAddr 返回实例的数据区地址
* 返回值  : 正确返回 S_OK, 其他返回则异常
*****************************************************************/
T_RESULT DataUnivIterNext(WORD32 * pdwInst, LPVOID * ppAreaAddr);
```

```
/*****************************************************************
* 函数名称: DataUnivIterFirst
* 功能描述: 查找数据区的第一个记录,带出索引和对应数据区的地址
* 输出参数: WORD32 * pdwInst 返回实例的数组的索引
*           LPVOID * ppAreaAddr 返回实例的数据区地址
* 返回值  : 正确返回 S_OK, 其他返回则异常
*****************************************************************/
T_RESULT DataUnivIterFirst(WORD32 * pdwInst, LPVOID * ppAreaAddr);

void  Perform_OMC_Report(void)
{
    T_RESULT            wGlbRslt;
    WORD32                dwNodeIdx;
    T_UNIV_HASH_KEY        tuHashKey;
    T_ConnListNode    *    ptConnNode = NULL;
    T_PERFORM_INFO_SUM     *    pPerformSum    = NULL;
    T_PERFORM_NODE_INFO     *    pPerformNodeInfo = NULL;

    wGlbRslt = DataGetGlbVar(GREG_RES_PERFORM_SUM, (LPVOID*)&
       pPerformSum);

    wGlbRslt  =    DataIterFirst(&dwNodeIdx, (LPVOID*)&
       pPerformNodeInfo);
    while (wGlbRslt    == S_OK    && pPerformNodeInfo    != NULL
           && dwNodeIdx !=    UNIV_UNUSED_WORD32)
    {
        if (pPerformNodeInfo->dwValid  == PERFORM_TRUE)
        {
            ptConnNode = NULL;
            tuHashKey.dwKey    = pPerformNodeInfo->dwValue;
            wGlbRslt = DataSearchByKey (&tuHashKey, &dwNodeIdx, (
               LPVOID*)&ptConnNode);
            if (ptConnNode != NULL && wGlbRslt == S_OK)
            {
                pPerformSum->dwRecvPPRNum +=ptConnNode->dwRecvPPRNum;
                pPerformSum->dwRecvRTRNum +=ptConnNode->dwRecvRTRNum;
            }
            else
            {
                ; //这个地方不需要特殊处理
            }
        }

        else
        {
            ; //这个地方不需要特殊处理
        }
```

```
        wGlbRslt  =    DataIterNext(&dwNodeIdx, (LPVOID*) &
           pPerformNodeInfo);
    }
    // ……省略
    return;
}
```

1.3 请找出下面代码中的隐患或者错误,说明故障原因并改正。(6 分)

```
char g_nmsbcReqBuf[MAX_REQ_SIZ];
char g_nmsbcAckBuf[MAX_REQ_SIZ];
#define RM_OAM_TEMP_BUF_LEN 256

WORD NM_SetMLIaInterfaceCmd(NM_ML_IA * ptMLIaInterface)
{
    DWORD        dwLeftDataLen = 0;
    BYTE        bTempBuf[RM_OAM_TEMP_BUF_LEN];
    BYTE         * pbTempBuf  = NULL;
    DWORD        dwTmpDataLen  = 0;
    MSG_COMM_OAM * pReqMsg      = (MSG_COMM_OAM *)g_nmsbcReqBuf;
    MSG_COMM_OAM * pAckMsg      = (MSG_COMM_OAM *)g_nmsbcAckBuf;
    NM_ML_IA     * ptMIa       = NULL;
    DWORD           dwLoopNum  = 0;

    if ( NULL == ptMLIaInterface)
    {
    return GEN_ERR;
}

pbTempBuf = bTempBuf;

for(dwLoopNum = 0; dwLoopNum < NM_IA_INTERFACE_MAX; dwLoopNum++)
{
    ptMIa = &ptMLIaInterface[dwLoopNum];
    if ( !ptMIa->blUsedFlag )
    {
        continue;
    }
    dwLeftDataLen = RM_OAM_TEMP_BUF_LEN - 1;
    dwTmpDataLen = snprintf( pbTempBuf, dwLeftDataLen,
        ”ip-address ipv4 %s port %u local domain %s transport ”,
        ptMIa->tLocalIA.bIpAddr, ptMIa->tLocalIA.wPort,
        ptMIa->tLocalIA.bIaDomain );

    pbTempBuf      += dwTmpDataLen;
    dwLeftDataLen -= dwTmpDataLen;/* 命令行的最大长度限制为255, */
                                  /* 所以这里不考虑减法溢出了    */
    dwTmpDataLen  = snprintf( pbTempBuf, dwLeftDataLen, ” udp” );
```

```
        bTempBuf[RM_OAM_TEMP_BUF_LEN] = ’\0’;
        if ( GEN_ERR == NM_ConstructOamMsg(bTempBuf, pReqMsg))
        {
            return GEN_ERR;
        }

        if ( SUCC_AND_NOPARA != NM_SendExecMsg(pReqMsg, pAckMsg))
        {
            return pAckMsg->ReturnCode;
        }
    }
    return NO_ERROR;
}
```

1.4 请找出下面代码中的隐患或者错误,说明故障原因并改正。(9 分)

```
/* 添加变更通知的注册表项 */
#define  DBS_NOTIFY_ADD_REG_ITEM(req,dbNames,tabName,type,event)   \
do{\
    if(req.wTableNum <= _DB_CFGCHG_REG_TABLE_NUM_MAX)\
    {\    strncpy(req.tTable[req.wTableNum].dbName,\
                dbNames,(_DB_NAME_LEN-1));\
        strncpy(req.tTable[req.wTableNum].tableName,\
               (CHAR *)tabName,(_DB_TABLE_NAME_MAX-1));\
        req.tTable[req.wTableNum].ucNotifyType    = type;\
        req.tTable[req.wTableNum].dwAppNotifyID = event;\
        req.wTableNum++;\
    }\
    else \
    {\
        XOS_SysLog(DBS_PRNLEVEL_ERROR,\
                  ”[DBS]: line %d: NOTIFY_ADD_REG_ITEM req.wTableNum
                      max,failed ! table[%s]\n”, \
        __LINE__, req.wTableNum, tabName );\
    }\
}while(0)

WORD32 DBS_APP_Notify_Reg_H248()
{
    WORD32    dwJIDNo;
    JID       tJIDNotify;
    XOS_STATUS  dwRet;
    _db_t_cfgchg_reg_req   tH248Item;

    memset(&tH248Item,0,sizeof(_db_t_cfgchg_reg_req));
    dwRet = XOS_GetSelfJID(&tH248Item.tJID);
    if (XOS_SUCCESS != dwRet)
    {
        return RC_ERROR;
```

```
    }
    tH248Item.ucFlag     = _DB_CFGCHG_NOTIFY_REG;
    tH248Item.ucPacketId = 0;

    DBS_NOTIFY_ADD_REG_ITEM(tH248Item,
                            PSS_CONFIG_DATABASE,
                            ”R_UDPPORT”,
                            _DB_CFGCHG_NOTIFY_TUPLE_TYPE,
                            EV_COMM_UDPPORTCHG);
    dwJIDNo = XOS_ConstructJNO(JOB_TYPE_DBS_NOTIFY,1);
    dwRet = XOS_GetJIDByJNO(dwJIDNo, &tJIDNotify);
    if (XOS_SUCCESS != dwRet)
    {
        return RC_ERROR;
    }

    dwRet = XOS_SendAsynMsg(EV_CFGCHG_REG_APP_TO_DBS_REQ,
                            (BYTE *)&tH248Item,
                            sizeof(_db_t_cfgchg_reg_ack),
                            OS_MSG_VER_DEFAULT,
                            XOS_MSG_HIGH, &tJIDNotify);
    if (XOS_SUCCESS != dwRet)
    {
        return RC_ERROR;
    }
    return RC_OK;
}
```

1.5 请找出下面代码中的隐患或者错误,说明故障原因并改正。(9 分)

```
BYTE  SdpDecodeAttribute_Qos( BYTE **ppMessage,
   SDP_PRECONDITION_INFO_t * ptQos )
{
    DWORD bLocation;
    BYTE  bAttr[60];
    BYTE  bCurrNum = 0; /* the number of total current status line*/
    BYTE  bDesNum  = 0; /* the number of total des status line*/
    BYTE  bConfNum = 0; /* the number of conf status line*/
    if( (ptQos == NULL) || (ppMessage == NULL))
    {
        return SDP_DEC_ERROR;
    }
    memset( ptQos, 0, sizeof(SDP_PRECONDITION_INFO_t) );
    while(**ppMessage != SDP_SIGN_ENDOFSTRING)
    {
        while((**ppMessage == SDP_SIGN_SPACE) || (**ppMessage ==
           SDP_SIGN_TAB))
            (*ppMessage)++;
        bLocation = 0;
        if (’a’ != **ppMessage)
        {
            continue;
        }
        (*ppMessage)++;
        if(**ppMessage != SDP_SIGN_EQUAL)
        {
            return(SDP_DEC_ERROR);
        }
        (*ppMessage)++; /* jump off the character ’=’ */
        /* space disallowed */
        if(**ppMessage == SDP_SIGN_SPACE)
        {
            return(SDP_DEC_ERROR);
        }
        else
        {
            while( (**ppMessage != SDP_SIGN_CARRIAGERETURN) &&
                   (**ppMessage != SDP_SIGN_LINEFEED) )
            {
                bAttr[bLocation] = **ppMessage;
                (*ppMessage)++;
                bLocation++;
                if(**ppMessage == SDP_SIGN_COLON)
                {
                    break;
                }
                if(**ppMessage == SDP_SIGN_ENDOFSTRING)
                {
                    return(SDP_DEC_SUCCESS);
                }
            }
            bAttr[bLocation] = SDP_SIGN_ENDOFSTRING;
            if((**ppMessage == SDP_SIGN_CARRIAGERETURN) ||
               (**ppMessage == SDP_SIGN_LINEFEED))
            {
                SdpReadLine(ppMessage); /* 跳到下一个 a 行*/
                continue;
            }
        }
    }
    return(SDP_DEC_SUCCESS);
}
```

1.6 请找出下面代码中的隐患或者错误,说明故障原因并改正。(8 分)

```
typedef struct tagOmcMsg
{
    BYTE    ucVer;     /* 消息头版本号 */
    BYTE    ucMsgVer;  /* 消息体版本号 */
    BYTE    ucMsgType;/* 消息类型 */
    BYTE    ucVerType;/* 前后台消息版本指, V3 或者 V4 版本*/
    BYTE    ucPad2;    /* 填充字节 2 */
    WORD16  wMsgSize;  /* 应用消息体大小 */
    WORD32  dwMsgId;   /* 应用消息 ID */
    WORD32  dwEndian;  /* 前台字节序 */
    JID     tSender;   /* 发送方 JID */
    JID     tReceiver;/* 接收方 JID */
} T_MSGOMC_HDR;

BYTE *MsgOmcMsgInvert(void *ptMsg)
{
    WORD32  dwHeadLen;
    T_MSGOMC_HDR *ptSrcMsg=NULL;
    BYTE    *pucDestMsg=NULL;
    /* 获取互斥信号量资源 */
    bl8Ret  = XOS_ProcessSemP(g_dwSemId,WAIT_FOREVER);
    if(!bl8Ret)
    {
        return NULL;
    }
    ptSrcMsg =( T_MSGOMC_HDR *)ptMsg;
    dwHeadLen = sizeof(T_MSGOMC_HDR);
    /* 重新计算本板对应版本的消息头大小和送来的消息体 */
    /* 动态申请内存 */
    pucDestMsg = XOS_GetUB(dwHeadLen + ptSrcMsg->wMsgSize);
    /* 拷贝消息头 */
    memcpy(pucDestMsg, ptSrcMsg, dwHeadLen);
    /* 整理消息体 */
    memcpy(pucDestMsg + dwHeadLen, ptSrcMsg +dwHeadLen, ptSrcMsg ->
        wMsgSize);
    /* 释放互斥信号量资源 */
    XOS_ProcessSemV(g_dwSemId);

    return pucDestMsg;
}
```

1.7 请找出下面代码中的隐患或者错误,说明故障原因并改正。(6 分)

```
RM_ERROR_NO RM_SG_GetSignalGroupInfo(
                    IN SIGNAL_GROUP_ACCESS_REQ *ptReq,
                    OUT SIGNAL_GROUP_ACCESS_ACK *ptAck )
{
    SIGNAL_GROUP   *ptSignalGroup          = NULL;
    NEXTHOP        *ptNextHopTemp          = NULL;
    NEXTHOP        *ptNextHopTrackTemp     = NULL;
    NEXTHOP        *ptNextHop              = NULL;
    NEXTHOP        *ptTrackNextHop         = NULL;
    NEXTHOP        *ptOtherNextHop         = NULL;
    NEXTHOP        *ptTrackOtherNextHop    = NULL;
    WORD            wRet                   = RC_OK;

    ptAck->bResult = FALSE;

    if ( NULL == ptReq || NULL == ptAck )
    {
        return RM_ERROR_NULL_POINTER_ID;
    }

    memset((BYTE *)ptAck, 0, sizeof(SIGNAL_GROUP_ACCESS_ACK));
    ptSignalGroup =
        RM_Signal_Group_GetSignalGroupByID( ptReq->wSignalGroupID );
    if ( NULL == ptSignalGroup )
    {
        return RM_ERROR_SIGNAL_GROUP_NOT_EXIST;
    }

    wRet = RM_SG_FindNextHopByID( ptSignalGroup->wTrackNextHop,
                                  &ptNextHopTemp,
                                  &ptNextHopTrackTemp);
    if ( RC_OK == wRet )
    {
        ptNextHop      = ptNextHopTemp;
        ptTrackNextHop = ptNextHopTrackTemp;
    }

    wRet = RM_SG_FindNextHopByID( ptSignalGroup->wTrackOtherNextHop,
                                  &ptNextHopTemp,
                                  &ptNextHopTrackTemp);
    if ( RC_OK == wRet )
    {
        ptOtherNextHop      = ptNextHopTemp;
        ptTrackOtherNextHop = ptNextHopTrackTemp;
    }

    ptAck->wNextHopID     = ptNextHop->wID;
    memcpy( (BYTE *)&ptAck->tNextHopIP,
            (BYTE *)&ptNextHop->tNextHopIp.tIpAddr,
            sizeof( RM_IPA ) );

    if ( NULL != ptTrackNextHop )
    {
```

```
        ptAck->wTrackNextHopID     = ptTrackOtherNextHop ->wID;
        memcpy( (BYTE *)&ptAck->tTrackNextHopIP,
                (BYTE *)& ptTrackOtherNextHop->tNextHopIp.tIpAddr,
                sizeof( RM_IPA ) );
    }

    if ( NULL != ptTrackOtherNextHop  &&  ptOtherNextHop != NULL )
    {
        ptAck->wOtherNextHopID     = ptOtherNextHop->wID;
        memcpy( (BYTE *)&ptAck->tOtherNextHopIP,
                (BYTE *)&ptOtherNextHop->tNextHopIp.tIpAddr,
                sizeof( RM_IPA ) );
        ptAck->wTrackOtherNextHopID = ptTrackOtherNextHop->wID;
        memcpy( (BYTE *)&ptAck->tTrackOtherNextHopIP,
                (BYTE *)& ptTrackOtherNextHop->tNextHopIp.tIpAddr,
                sizeof( RM_IPA ) );
    }

    ptAck->bResult = TRUE;

    return RM_ERROR_NO_ERROR;
}
```

1.8 请找出下面代码中的隐患或者错误,说明故障原因并改正。(9 分)

```
int filecpy(char* pFileFrom, char* pFileTo)
{
    int iFileLenth=0;
    char* pBuf=NULL;
    FILE* fpFileFrom,*fpFileTo;
    if(strlen(pFileFrom)==0  || strlen(pFileTo)==0)
    {
        return ERROR;
    }
    if (pFileFrom != NULL|| !pFileTo!= NULL)
    {
        return ERROR;
    }
    fpFileFrom=fopen(pFileFrom,"rb");
    if(fpFileFrom==NULL)
    {
        return ERROR;
    }
    fseek(fpFileFrom,0,SEEK_END);
    iFileLenth =ftell(fpFileFrom);
    if(iFileLenth>0)
    {
        pBuf=(char*)malloc(iFileLenth);
        if(!pBuf)
        {
            return ERROR;
        }
    }
    fseek(fpFileFrom,0,SEEK_SET);
    fread(pBuf,iFileLenth,1,fpFileFrom);
    fpFileTo=fopen(pFileTo,"w+");
    if(fpFileTo==NULL)
    {
        return ERROR;
    }
    fwrite(pBuf,iFileLenth,1,fpFileTo);
    fclose(fpFileFrom);
    fclose(fpFileTo);
    free(pBuf);
    return 0;
}
```

2 填空题 (共 20 分)。

2.1 下列程序运行后的输出结果是 _____,_____。(2 分)

```
void main()
{
    char a[7]= "a0\\\0a0\0";
    int i,j;
    i=sizeof(a);
    j=strlen(a);
    printf("%d ,%d\n",i,j);
}
```

2.2 下列程序运行后的输出结果是 _____,_____。(2 分)

```
void main()
{
    int a[5] = {0,1,2,3,4};
    int *ptr = (int *)(&a+1);
    printf("%d,%d", *(a+1), *(ptr-1));
}
```

2.3 定义整型变量 int a;, 按下列要求写出赋值表达式 (说明:整数 a 从最低位到最高位,依次为第 1 到 32 位):(2 分)

将 a 的第 2 位和第 5 位置为 1, 其它的值不变 _____。

将 a 的第 2 位和第 5 位置为 0, 其它的值不变 _____。

2.4 不考虑内存申请失败的情况,则下列程序的运行结果是 _____。(1 分)

```
void main()
{
    char *p1 = "name";
    char *p2 = NULL;
    p2 = (char*)malloc(20);
    memset (p2, 0, 20);
    while(*p2++ = *p1++);
    printf("%s\n",p2);
    free(p2);
}
```

2.5 strlen(""); strlen(NULL); strlen("0"); sizeof(0);执行结果分别为: _____,_____,_____,_____。(4 分)

2.6 设有:int a=10,b=20,c=30,d=40,m=50,n=60;执行(m=a>b)&&(n=c>d)后 n 的值为 _____。(1 分)

2.7 下列程序运行后的输出结果是 _____,_____。(2 分)

```
void getSubSystems (int i,char* pName)
{
    char* pSystems[4] ={"SIG","H248","DB","SIP"};
    pName= pSystems[i];
    return;
}
void GetYourName(char ** ptrStr)
{
    static char sString[] = "XXX";
    *ptrStr = sString;
}
int main(int argc, char* argv[])
{
    char* pName="YYY";
    getSubSystems(3,pName);
    printf("%s   \n",pName);
    pName = NULL;
    GetYourName(&pName);
    printf("%s   \n",pName);
    return 0;
}
```

2.8 下列程序运行后的输出结果是 _____,_____,_____。(3 分)

```
#define IPV4_MAX 20

char * iptostring(unsigned char *Ipv4Addr)
{
    static char bTemp[IPV4_MAX];
    snprintf( bTemp, IPV4_MAX,
              "%d.%d.%d.%d",
              Ipv4Addr[0],
              Ipv4Addr[1],
              Ipv4Addr[2],
              Ipv4Addr[3]);
    return bTemp;
}
void main()
{
    char   tOuterIP1[4] = {10,33,32,21};
    char   tOuterIP2[4] = {10,44,32,21};
    char   tOuterIP3[4] = {10,55,32,21};
    printf("%s,%s,%s",
           iptostring(&tOuterIP1),
           iptostring(&tOuterIP2),
           iptostring(&tOuterIP3));
}
```

2.9 性能提升的方法主要有:(3 分)

_____,

_____,

_____。

3　编程题 (共 20 分)。

3.1　给定单链表头结点,删除链表中倒数第 k 个结点。(10 分)

节点定义为:

```
struct NODE
{
    int nValue;
    struct NODE * next;
};
```

最后一个节点的next = NULL。

函数原型:NODE * DeleteNode(NODE * pList,int k);

返回值: 链表头节点。

3.2　设计一个算法将字符串对调,不能借用其他存储空间。(10 分)

函数原型:bool ConverseString(char * chStr);chStr指向待转换的字符串。

返回值: 转换是否成功。

举例: 字符串chStr = "abcdefg"转换后变为chStr = "gfedcba"。