

Paradigmas de Programação I

Teste Prático 2B

Nome: _____ N° Mec.: _____

1. Considere a classe chamada CString para manipulação de um array de caracteres. Para resolver os exercícios propostos é-lhe permitido, sempre que achar necessário, usar apenas as seguintes funções da biblioteca string.h:

```
int strcmp( const char *string1, const char *string2 );
unsigned int strlen( const char *string );
char *strcpy( char *strDestination, const char *strSource );
```

```
class CString
{
public:
    CString();
    CString(const char c, int replicas);
    virtual ~CString();

    int Count(char c);
    void ToggleChar();
    void operator-();
    int operator[](int index);

    char* m_pString;
};

CString::CString()
{
    m_pString = NULL;
}
```

Dado o construtor por defeito, implemente os seguintes métodos:

- 1.1. O construtor **CString(const char c, int replicas)**, que deverá construir uma string constituída pelo caracter c repetido um determinado n° de vezes dado pela variável replicas, e o destrutor.

- 1.2. O método **int Count(char c)** para devolver o número de ocorrências do caracter c. Caso o caracter não pertença à string, a função deve devolver -1.

1.3. O método `void ToggleChar()` para alterar todas as minúsculas para maiúsculas e vice-versa. Pode utilizar as funções `bool islower(char c)` e `bool isupper(char c)` que devolvem verdadeiro quando o carácter é minúsculo ou maiúsculo, respectivamente, e as funções `char tolower(char c)` e `char toupper(char c)` para devolver o carácter c convertido em minúscula ou maiúscula, respectivamente.

1.4. O método `void operator-()` para apagar da string todos os algarismos.

1.5. O método `int operator[](int index)` para devolver o carácter armazenado na string `m_pString` na posição `index`. Em caso de erro a função deverá devolver o valor -1.

2. Considere a lista ligada CLinkedList semelhante à que implementou nas aulas práticas. Dado o método InsertHead para inserção de um elemento à cabeça da lista, implemente um método, chamado InsertTail para inserir um elemento no final da lista.

```

class CLinkedList
{
...
public:
    CLinkedList();
    virtual ~CLinkedList();
public:
    int InsertHead(const void* pData,
                  unsigned int dataSize);
    int RemoveHead(void* pData,
                  unsigned int dataSize);
    int InsertTail(const void* pData,
                  unsigned int dataSize);
    const void* GetHead();
    const void* GetNext();
    unsigned int Count();
...
protected:
    unsigned int m_count;
    SNode* m_pHead;
    SNode* m_pCurrent;
};

int CLinkedList::InsertHead(const void* pData,
                          unsigned int dataSize)
{
    SNode* pNode;

    if (pData == NULL)
        return NULL_PTR;
    pNode = new SNode;
    if (pNode == NULL)
        return OUT_OF_MEMORY;
    pNode->m_pData = new char[dataSize];
    if (pNode->m_pData == NULL)
    {
        delete pNode;
        return OUT_OF_MEMORY;
    }
    memcpy(pNode->m_pData, pData, dataSize);
    pNode->m_pNext = m_pHead;
    m_pHead = pNode;
    ++m_count;
    return OK;
}

```

3. Dado o programa, responda às seguintes questões:

```
int main(int argc, char* argv[])
{
    CString str1, str2('p', 4), str3('a', 3);
    CLinkedList list;

    int error1 = list.InsertHead(str1.m_pString, strlen(str1.m_pString)+1);
    int error2 = list.InsertHead(str2.m_pString, strlen(str2.m_pString)+1);

    cout << list.Count() << endl;
    cout << (char*)list.GetHead() << endl;
    cout << ((char*)list.GetHead())[0] << endl;
    cout << *(char*)list.GetHead() << endl;
    cout << *((char*)list.GetHead()+2) << endl;

    list.InsertTail(str3.m_pString, strlen(str3.m_pString)+1);
    list.RemoveHead(str2.m_pString, strlen(str2.m_pString)+1);
    cout << (char*)list.GetHead() << endl;
    return 0;
}
```

3.1. Qual o valor de **error1** e **error2** assumindo que as operações de alocação de memória são sempre bem sucedidas.

3.2. Escreva o resultado deste programa tal como apareceria no ecrã do seu computador.