1．编写一个 C 函数，该函数在一个字符串中找到可能的最长的子字符串，且该字符串是由同一字符组成的。

```
char* search(char* cpSource, char ch)
{
     if(cpSource==NULL)
          return 0;
     char*cpTemp=NULL, *cpDest=NULL;
     int iTemp, iCount=0;
     while(*cpSource)
     {
          if(*cpSource == ch)
          {
               iTemp = 0;
               cpTemp = cpSource;
               while(*cpSource == ch)
                    ++iTemp, ++cpSource;
               if(iTemp > iCount)
                    iCount = iTemp, cpDest = cpTemp;
               if(!*cpSource)
                    break;
          }
          ++cpSource;
     }
     return cpDest;
}
```

2．请编写一个 C 函数，该函数在给定的内存区域搜索给定的字符，并返回该字符所在位置索引值。

```
int search(char*cpSource, char ch)
{
    if(cpSource==NULL)
        return 0;
    int i;
    int n=strlen(cpSource);
    for(i=0; i<n; i++)
        if(ch == cpSource[i])
            return i;
}
```

3.写一个函数比较两个字符串 str1 和 str2 的大小，若相等返回 0，若 str1 大于 str2 返回 1，若 str1 小于 str2 返回－1

```
int strcmp ( const char *dst,const char * src)
{
    int ret = 0 ;
    while( ! (ret = *(unsigned char *)src - *(unsigned char *)dst) && *dst++&&*src++);
    if ( ret < 0 )
        ret = -1 ;
    else if ( ret > 0 )
        ret = 1 ;
    return( ret );
}
```

4.求 1000! 的末尾有几个 0
每个 0 拆成 2*5 的形式，因为 2*5 会参生一个 0，例：90=2*5*9.因总的 2 因子很多，所以 0 的个数，由 5 因子个数决定，即等于 5 因子个数.求出 1->1000 里,能被 5 整除的数的个数 n1,能被 25 整除的数的个数 n2,能被 125 整除的数的个数 n3,能被 625 整除的数的个数 n4.1000!末尾的零的个数=n1+n2+n3+n4;

```
int find5(int num)
{
    int ret=0;
    while(num%5==0)
    {
        num/=5;
        ret++;
    }
    return ret;
}
int main()
{
    int result=0;
    int i;
    for(i=5;i<=NUM;i+=5)
    {
        result+=find5(i);
    }
    printf(" the total zero number is %d\n",result);
    return 0;
}
```

5．有双向循环链表结点定义为：

```
struct node
{
    int data;
    struct node *front,*next;
};
```

有两个双向循环链表 A，B，知道其头指针为：pHeadA,pHeadB，请写一函数将两链表中 data 值相同的结点删除

```
BOOL DeteleNode(Node *pHeader, DataType Value)
{
    if (pHeader == NULL)
        return;
    BOOL bRet = FALSE;
    Node *pNode = pHead;
    while (pNode != NULL)
    {
        if (pNode->data == Value)
        {
            if (pNode->front == NULL)
            {
                pHeader = pNode->next;
                pHeader->front = NULL;
            }
            else
            {
                if (pNode->next != NULL)
                {
                    pNode->next->front = pNode->front;
                }
                pNode->front->next = pNode->next;
            }
            Node *pNextNode = pNode->next;
            delete pNode;
            pNode = pNextNode;
            bRet = TRUE;
            //不要break或return, 删除所有
        }
        else
        {
            pNode = pNode->next;
        }
    }
    return bRet;
}
void DE(Node *pHeadA, Node *pHeadB)
{
    if (pHeadA == NULL || pHeadB == NULL)
    {
        return;
    }
    Node *pNode = pHeadA;
    while (pNode != NULL)
    {
        if (DeteleNode(pHeadB, pNode->data))
        {
            if (pNode->front == NULL)
            {
                pHeadA = pNode->next;
                pHeadA->front = NULL;
            }
            else
            {
                pNode->front->next = pNode->next;
                if (pNode->next != NULL)
                {
                    pNode->next->front = pNode->front;
                }
            }
            Node *pNextNode = pNode->next;
            delete pNode;
            pNode = pNextNode;
        }
        else
        {
            pNode = pNode->next;
        }
    }
}
```

6．编程实现：找出两个字符串中最大公共子字符串,如"abccade","dgcadde"的最大子串为"cad"

```
int GetCommon(char *s1, char *s2, char **r1, char **r2)
{
    int len1 = strlen(s1);
    int len2 = strlen(s2);
    int maxlen = 0;
    for(int i = 0; i < len1; i++)
    {
        for(int j = 0; j < len2; j++)
        {
            if(s1[i] == s2[j])
            {
                int as = i, bs = j, count = 1;
                while(as + 1 < len1 && bs + 1 < len2 && s1[++as] == s2[++bs])
                    count++;
                if(count > maxlen)
                {
                    maxlen = count;
                    *r1 = s1 + i;
                    *r2 = s2 + j;
                }
            }
        }
    }
```

7．编程实现：把十进制数(long 型)分别以二进制和十六进制形式输出，不能使用 printf 系列库函数

```
char* test3(long num)
{
    char* buffer = (char*)malloc(11);
    buffer[0] = '0';
    buffer[1] = 'x';
    buffer[10] = '\0';
    char* temp = buffer + 2;
    for (int i=0; i < 8; i++) {
        temp[i] = (char)(num<<4*i>>28);
        temp[i] = temp[i] >= 0 ? temp[i] : temp[i] + 16;
        temp[i] = temp[i] < 10 ? temp[i] + 48 : temp[i] + 55;
    }
    return buffer;
}
```

8.输入 N, 打印 N*N 矩阵
比如 N = 3，打印：

```
1 2 3
8 9 4
7 6 5
```

N = 4，打印：

```
1 2 3 4
12 13 14 5
11 16 15 6
10 9 8 7
```

解答：

1
```
#define N 15
int s[N][N];
void main()
{
    int k = 0, i = 0, j = 0;
    int a = 1;
    for( ; k < (N+1)/2; k++ )
    {
        while( j < N-k ) s[i][j++] = a++; i++; j--;
        while( i < N-k ) s[i++][j] = a++; i--; j--;
        while( j > k-1 ) s[i][j--] = a++; i--; j++;
        while( i > k ) s[i--][j] = a++; i++; j++;
    }
    for( i = 0; i < N; i++ )
    {
        for( j = 0; j < N; j++ )
            cout << s[i][j] << '\t';
        cout << endl;
    }
}
```

2
```
#define MAX_N 100
int matrix[MAX_N][MAX_N];

void SetMatrix(int x, int y, int start, int n) {
    int i, j;
    if (n <= 0) //递归结束条件
        return;
    if (n == 1) { //矩阵大小为时
        matrix[x][y] = start;
        return;
    }
    for (i = x; i < x + n-1; i++) //矩阵上部
```

```
        matrix[y][i] = start++;
    for (j = y; j < y + n-1; j++) //右部
        matrix[j][x+n-1] = start++;
    for (i = x+n-1; i > x; i--) //底部
        matrix[y+n-1][i] = start++;
    for (j = y+n-1; j > y; j--) //左部
        matrix[j][x] = start++;
    SetMatrix(x+1, y+1, start, n-2); //递归
}
void main() {
    int i, j;
    int n;
    scanf("%d", &n);
    SetMatrix(0, 0, 1, n);
    //打印螺旋矩阵
    for(i = 0; i < n; i++) {
        for (j = 0; j < n; j++)
            printf("%4d", matrix[i][j]);
        printf("\n");
    }
}
```

9．斐波拉契数列递归实现的方法如下：

```
int Funct( int n )
{
    if(n==0) return 1;
    if(n==1) return 1;
    return Funct(n-1) + Funct(n-2);
}
```

请问，如何不使用递归，来实现上述函数？

解答：

```
int Funct( int n ) // n  为非负整数
{
    int a=0;
    int b=1;
    int c;
    if(n==0) c=1;
    else if(n==1) c=1;
    else for(int i=2;i<=n;i++) //应该n从开始算起
    {
        c=a+b;
        a=b;
        b=c;
    }
    return c;
}
```

10．判断一个字符串是不是回文

```
int IsReverseStr(char *aStr)
{
	int i,j;
	int found=1;
	if(aStr==NULL)
		return -1;
	j=strlen(aStr);
	for(i=0;i<j;i++)
	{
		if(*(aStr+i)!=*(aStr+j-i-1)
		{
			found=0;
			break;
		}
	}
	return found;
}
```

11.Josephu 问题为：设编号为 1，2，… n 的 n 个人围坐一圈，约定编号为 k（1<=k<=n）的人从 1 开始报数，数到 m 的那个人出列，它的下一位又从 1 开始报数，数到 m 的那个人又出列，依次类推，直到所有人出列为止，由此产生一个出队编号的序列。

数组实现：

```
#include
#include
int Josephu(int n, int m)
{
    int flag, i, j = 0;
    int *arr = (int *)malloc(n * sizeof(int));
    for (i = 0; i < n; ++i)
        arr[i] = 1;
    for (i = 1; i < n; ++i)
    {
        flag = 0;
        while (flag < m)
        {
            if (j == n)
                j = 0;
            if (arr[j])
                ++flag;
            ++j;
        }
        arr[j - 1] = 0;
        printf("第%4d个出局的人是：%4d号\n", i, j);
    }
    free(arr);
    return j;
}
int main()
{
    int n, m;
    scanf("%d%d", &n, &m);
    printf("最后胜利的是%d号！\n", Josephu(n, m));
    system("pause");
    return 0;
}
```

链表实现：

```
#include
#include
```

```c
typedef struct Node
{
	int index;
	struct Node *next;
}JosephuNode;
int Josephu(int n, int m)
{
	int i, j;
	JosephuNode *head, *tail;
	head = tail = (JosephuNode *)malloc(sizeof(JosephuNode));
	for (i = 1; i < n; ++i)
	{
		tail->index = i;
		tail->next = (JosephuNode *)malloc(sizeof(JosephuNode));
		tail = tail->next;
	}
	tail->index = i;
	tail->next = head;
	for (i = 1; tail != head; ++i)
	{
		for (j = 1; j < m; ++j)
		{
			tail = head;
			head = head->next;
		}
		tail->next = head->next;
		printf("第%4d个出局的人是：%4d号\n", i, head->index);
		free(head);
		head = tail->next;
	}
	i = head->index;
	free(head);
	return i;
}
int main()
{
	int n, m;
	scanf("%d%d", &n, &m);
	printf("最后胜利的是%d号！\n", Josephu(n, m));
```

```
    system("pause");
    return 0;
}
```

10.已知strcpy函数的原型是：

```
char * strcpy(char * strDest,const char * strSrc);
```

1.不调用库函数，实现strcpy函数。

```
char * strcpy(char * strDest,const char * strSrc)
{
    ASSERT((strDest != NULL)&&(strSrc != NULL));
    char *strBuffer;
    strBuffer = strDest;
    while((*strDest++ = *strSrc++)!='\0');
    return strBuffer;
}
```

2.解释为什么要返回char *。

实现链式表达。