Tutorial 4 (Week 5)
Character Strings

Q1 (processString)

Write a C function that accepts a string str and returns the total number of vowels totVowels and digits totDigits in that string to the caller via call by reference. The function prototype is given as follows:

```
void processString(char *str, int *totVowels, int
*totDigits);
```

Write a C program to test the function.

Sample input and output sessions:

```
Test Case 1:
Enter the string:
I am one of the 400 students in this class.
Total vowels = 11
Total digits = 3

Test Case 2:
Enter the string:
I am a boy.
Total vowels = 4
Total digits = 0

Test Case 3:
Enter the string:
1 2 3 4 5 6 7 8 9
Total vowels = 0
Total digits = 9

Test Case 4:
Enter the string:
ABCDE
Total vowels = 2
Total digits = 0
```

Q1 (processString)

```
#include <stdio.h>
#include <string.h>
void processString(char *str, int *totVowels, int
*totDigits);
int main()
{
   char str[50], *p;
   int totVowels, totDigits;

   printf("Enter the string: \n");
   fgets(str, 50, stdin);
   if (p=strchr(str,'\n')) *p = '\0';
   processString(str, &totVowels, &totDigits);
   printf("Total vowels = %d\n", totVowels);
   printf("Total digits = %d\n", totDigits);
   return 0;
}
```

Version 1:

Q1 (processString)

```
void processString(char *str, int *totVowels, int
*totDigits)
{
   int i, size;
   *totVowels=0;
   *totDigits=0;
   i=0; size=0;
   while (str[i]!='\0'){
      size++;
      i++;
   }
   for (i=0; i < size; i++) {
      if (str[i] == 'a' || str[i] == 'e' ||
            str[i] == 'i' ||  str[i] == 'o' ||
             str[i] == 'u' || str[i] == 'A' ||
             str[i] == 'E' || str[i] == 'I' ||
             str[i] == 'O' || str[i] == 'U')
         (*totVowels)++;
      else if ( str[i] >= '0' &&  str[i] <= '9')
         (*totDigits)++;
   }
}
```

Version 2:

Q1 (processString)

```
void processString2(char *str, int *totVowels, int
*totDigits)
{
   int i,size;

   *totVowels = 0, *totDigits = 0;
   i=0; size=0;
   while (str[i]!='\0'){
      size++;
      i++;
   }
   for (i=0; i < size; i++) {
      if (*(str+i) == 'a' || *(str+i) == 'e' ||
            *(str+i) == 'i' || *(str+i) == 'o' ||
            *(str+i) == 'u' || *(str+i) == 'A' ||
            *(str+i) == 'E' || *(str+i) == 'I' ||
            *(str+i) == 'O' || *(str+i) == 'U')
         (*totVowels)++;
      else if ( *(str+i) >= '0' &&  *(str+i) <= '9')
         (*totDigits)++;
  }
}
```

Q2 (stringncpy)

Write a C function stringncpy() that copies not more than *n* characters (characters that follow a null character are not copied) from the array pointed to by *s2* to the array pointed to by *s1*.

If the array pointed to by *s2* is a string shorter than *n* characters, null characters are appended to the copy in the array pointed to by *s1*, until *n* characters in all have been written.

The stringncpy() returns the value of *s1*.

The function prototype is:

```
char *stringncpy(char * s1, char * s2, int n);
```

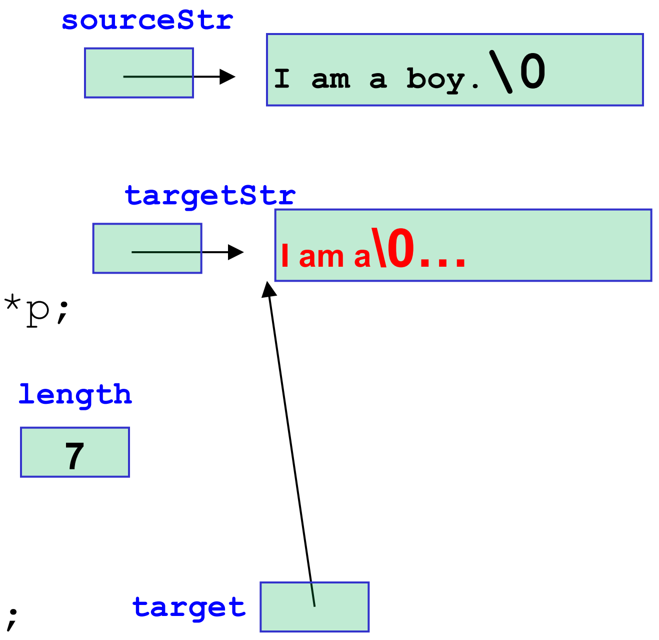
Write a C program to test the function.

Sample input and output sessions:

```
(1) Test Case 1
Enter the string:
I am a boy.
Enter the number of characters:
7
stringncpy(): I am a

(2) Test Case 2
Enter the string:
I am a boy.
Enter the number of characters:
21
stringncpy(): I am a boy.
```

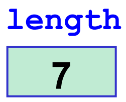
Q2 (stringncpy)

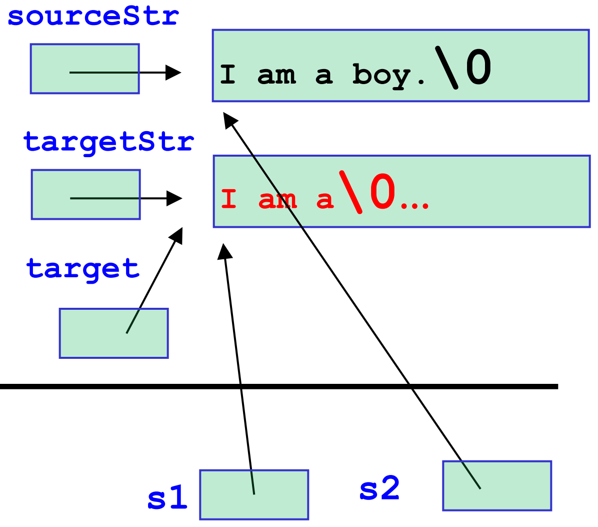
```
#include <stdio.h>
#include <string.h>
char *stringncpy(char *s1, char *s2, int n);
int main()
{
   char targetStr[40], sourceStr[40], *target, *p;
   int length;

   printf("Enter the string: \n");
   fgets(sourceStr, 40, stdin);
   if (p=strchr(sourceStr,'\n')) *p = '\0';
   printf("Enter the number of characters: \n");
   scanf("%d", &length);
   target = stringncpy(targetStr, sourceStr, length);
   printf("stringncpy(): %s\n", target);
   return 0;
}
```



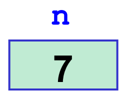
Q2 (stringncpy)

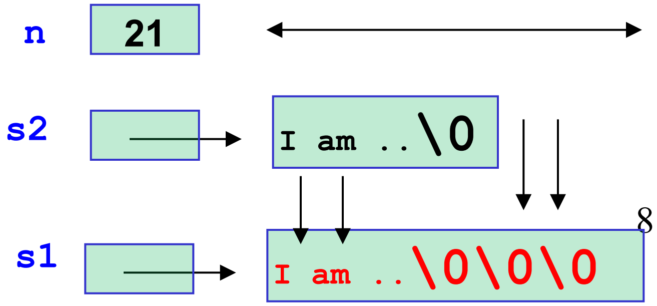
```
#include <stdio.h>
char *stringncpy(char *s1, char *s2, int n);
int main()
{
 target = stringncpy(targetStr, sourceStr, length);



}
```





```
char *stringncpy(char *s1, char *s2, int n){
   int k, h;
   for (k = 0; k < n; k++){
      if (s2[k] != '\0')
         s1[k] = s2[k];
      else
         break;
   }
   s1[k] = '\0';
   // to append '\0' after copying if s2 length is shorter than n
   for (h = k; h < n; h++)
      s1[h] = '\0';
   return s1;
}
```





Note: the last for loop in the code will not affect the correctness of the program; it only follows the question specification.

Q3 (stringcmp)

Write a C function that compares the string pointed to by *s1* to the string pointed to by *s2*.
If the string pointed to by *s1* is greater than, equal to, or less than the string pointed to by *s2*, then it returns 1, 0 or –1 respectively.
Write the code for the function without using the standard C string library function strcmp().

The function prototype is given as follows:

```
int stringcmp(char *s1, char *s2);
```

Write a C program to test the function.

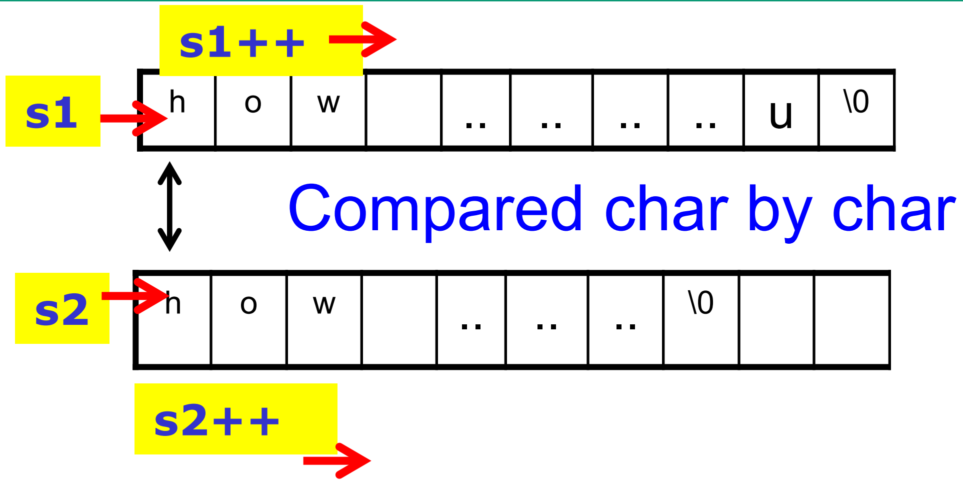
Sample input and output sessions:

```
Test Case 1:
Enter a source string:
abc
Enter a target string:
abc
stringcmp(): equal

Test Case 2:
Enter a source string:
abcdefg
Enter a target string:
abcde123
stringcmp(): greater than

Test Case 3:
Enter a source string:
abc123
Enter a target string:
abcdef
stringcmp(): less than
```

Q3 (stringcmp)

```
##include <stdio.h>
#include <string.h>
#define INIT_VALUE 999
int stringcmp(char *s1, char *s2);
int main()
{
   char source[80], target[80], *p;
   int result = INIT_VALUE;
   printf("Enter a source string: \n");
   fgets(source, 80, stdin);
   if (p=strchr(source,'\n')) *p = '\0';
   printf("Enter a target string: \n");
   fgets(target, 80, stdin);
   if (p=strchr(target,'\n')) *p = '\0';
   result = stringcmp(source, target);
   if (result == 1)
      printf("stringcmp(): greater than");
   else if (result == 0)
      printf("stringcmp(): equal");
   else if (result == -1)
      printf("stringcmp(): less than");
   else
      printf("stringcmp(): error");
   return 0;
}
```

```
int stringcmp(char *s1, char *s2){
    while (1) {
        if (*s1 == '\0' && *s2 == '\0')
            return 0;
        else if (*s1 == '\0')
            return -1;
        else if (*s2 == '\0')
            return 1;
        else if (*s1 < *s2)
            return -1;
        else if (*s1 > *s2)
            return 1;
        s1++;
        s2++;
    }
}
```



Comparison is based on ASCII value

The strcmp() Function: String Comparison based on ASCII Values

	0	1	2	3	4	5	6	7	8	9
0	NUL							BEL	BS	TAB
1	LF		FF	CR						
2								ESC		
3			SP	!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	'	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	DEL		

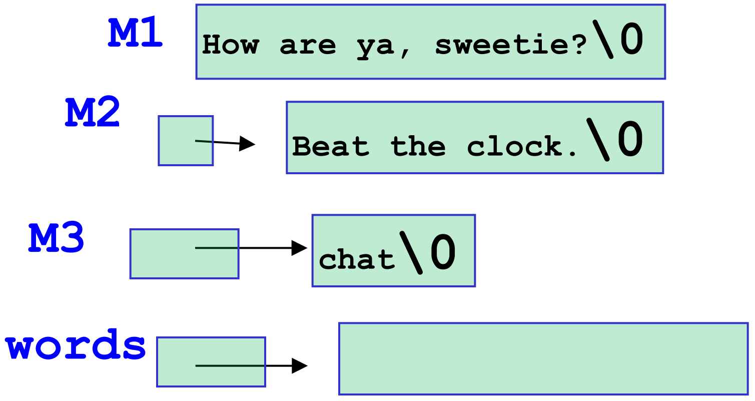
What does the following program print?

Q4

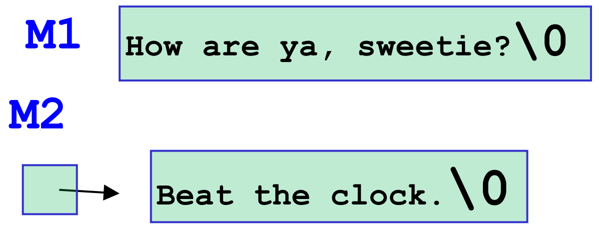
```
#include  <stdio.h>
#include  <string.h>
#define  M1  "How are ya, sweetie?"
char M2[40] = "Beat the clock.";
char *M3 = "chat";
int main()
{
   char words[80],*p;
   printf(M1);
   puts(M1);
   puts(M2);
   puts(M2+1);
   fgets(words, 80, stdin);   /* user inputs :  win a toy. */
   if (p=strchr(words,'\n')) *p = '\0';
   puts(words);
   scanf("%s", words+6);   /* user inputs :  snoopy. */
   puts(words);
   words[3] = '\0';
   puts(words);
   while (*M3)  puts(M3++);
   puts(--M3);
   puts(--M3);
   M3 = M1;
   puts(M3);
   return 0;
}
```

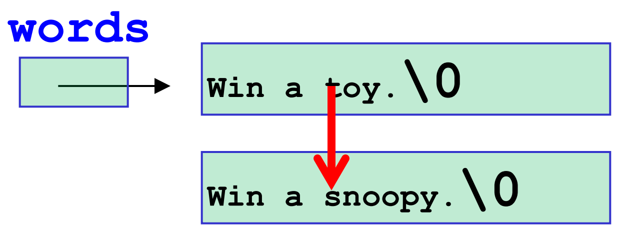
Note: A very good example to illustrate the processing of character strings. Please trace the code, and determine the output of the code.

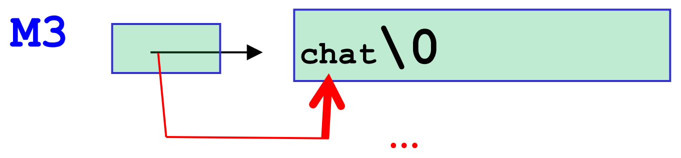
```c
#include  <stdio.h>
#define  M1  "How are ya, sweetie?"
char M2[40] = "Beat the clock.";
char *M3 = "chat";
int main(){
   char words[80];
   printf(M1);
   puts(M2);
   puts(M2+1);
   gets(words);   /* user inputs :  win a toy. */
   puts(words);
   scanf("%s", words+6); /* user inputs :  snoopy. */
   puts(words);
   words[3] = '\0';
   puts(words);
   while (*M3)
     puts(M3++);
   puts(--M3);
   puts(--M3);
   M3 = M1;
   puts(M3);
   return 0;
}
```



```c
#include  <stdio.h>
#define  M1  "How are ya, sweetie?"
char M2[40] = "Beat the clock.";
char *M3 = "chat";
int main(){
   char words[80];
   printf("%s",M1);
   puts(M2);
   puts(M2+1);
   gets(words);   /* user inputs :  win a toy. */
   puts(words);
   scanf("%s", words+6); /* user inputs :  snoopy. */
   puts(words);
   words[3] = '\0';
   puts(words);

   while (*M3)
     puts(M3++);
   puts(--M3);
   puts(--M3);

   M3 = M1;
   puts(M3);
   return 0;
}
```







Output

```
How are ya, sweetie?Beat the clock.
eat the clock.
win a toy.
win a toy.
snoopy.
win a snoopy.
win
chat
hat
at
t
t
at
How are ya, sweetie?
```