Tutorial 4 – Character Strings – Suggested Answers

1. What does the following program print?

```
#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;
}
```

Suggested answer: The program prints:

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

2. The following unknown function receives a string argument and a character argument, modifies the string argument and returns an integer value. Describe the purpose of the function. Give an example to support your answer.

```
int unknown(char str[ ], char c)
{
    int x, y=0, z=0;

    for (x=0; str[x] != '\0'; x++)
        if (str[x] != c)
            str[y++] = str[x];
        else
            z++;
    str[y] = '\0';
    return z;
}
```

Suggested answer:

The function removes char from the string to form a new string. At the same time, the number of characters in the string is returned to the calling function.
Example: str = " This is a string." & char = 's', then after executing the function, str = "Thi i a tring" & z = 3 will be returned.

3. Write the function strncpy() 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 strncpy returns the value of s1. The function prototype is:

    char *strncpy(char * s1, char * s2, int n);

Write a C program to test the function.

Suggested answer:

```
#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;
}
```

```
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';
   for (h = k; h < n; h++)
      s1[h] = '\0';
   return s1;
}
```

4. 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.

   Suggested answer:

```
#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++;
    }
}
```