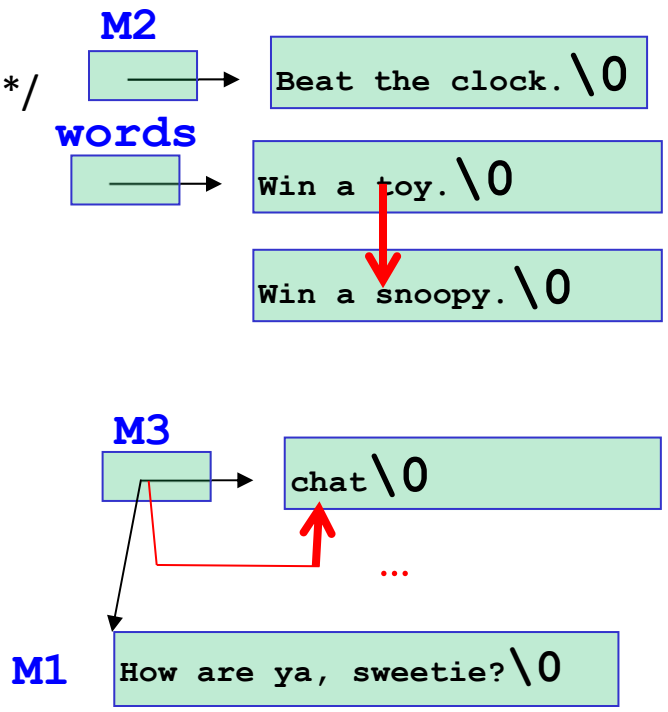
Tutorial 4 - Character Strings Q1

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;
}
```

Q1 – Suggested Answers

```
#include  <stdio.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(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;
}
```



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?

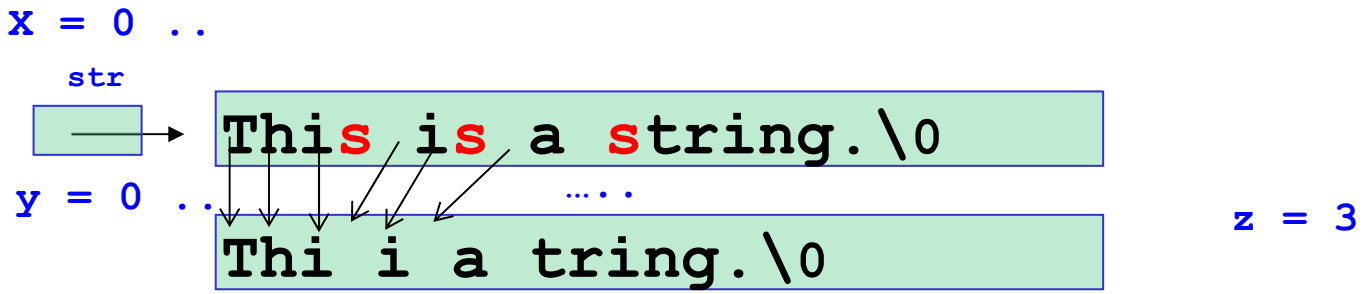
Q2

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;
}
```

Consider the **example**:
str = **"This is a string."** & char = **'s'**

Q2 – Suggested Answers




```
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;
}
```

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.

Q3 (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.

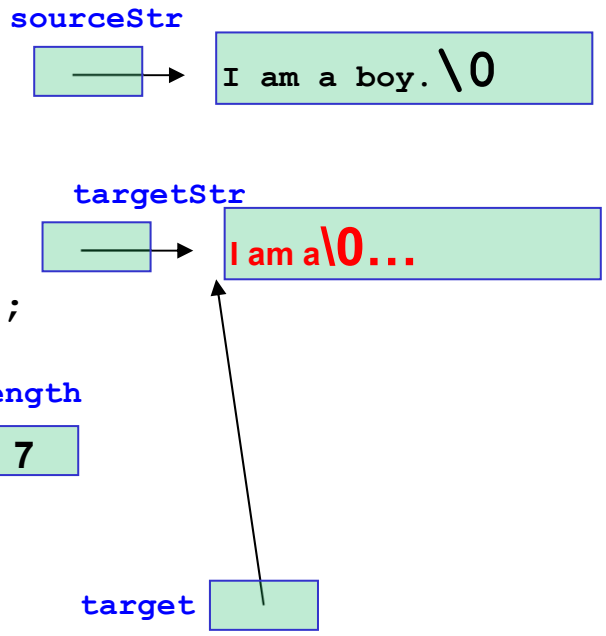
Sample input and output sessions:

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

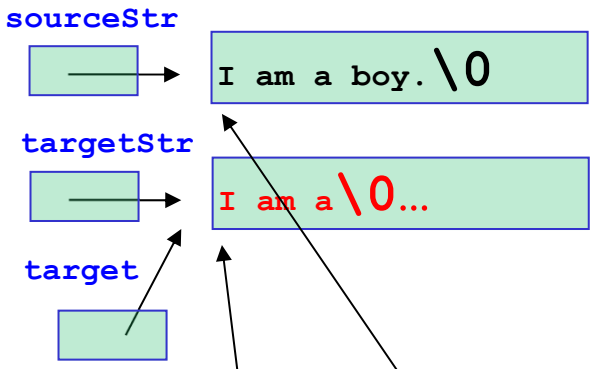
(2) Test Case 2
Enter the string:
I am a boy.
Enter the number of characters:
21
stringncpy(): I am a boy.

Q3 – 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;
}
```



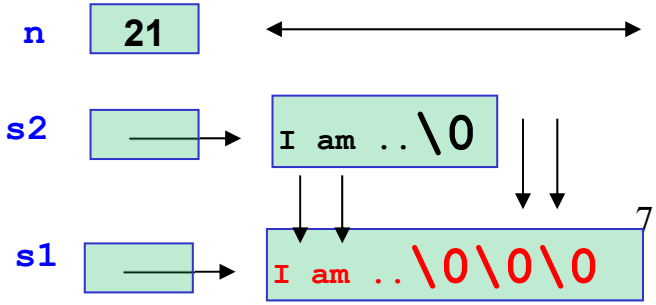
```
#include <stdio.h>
char *stringncpy(char *s1, char *s2, int n);
int main()
{
 target = stringncpy(targetStr, sourceStr, length);

}
```




```
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;
}
```

n 7

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



Q4 (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

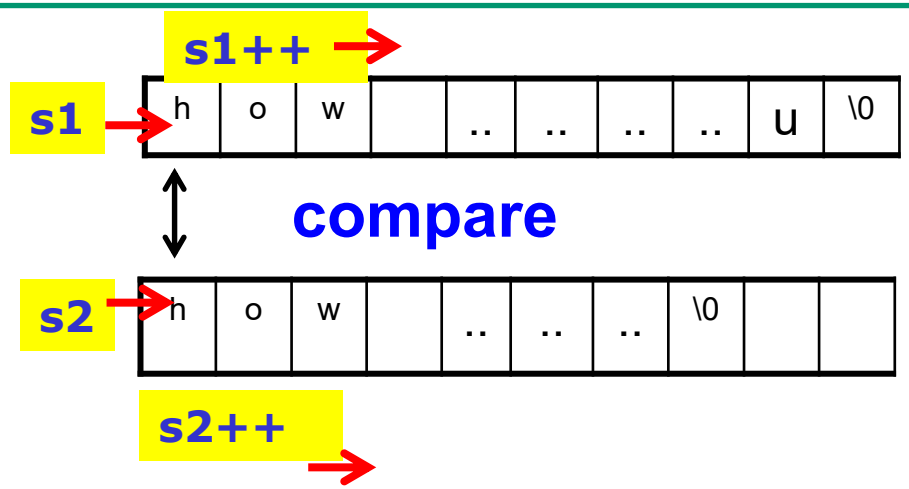
Test Case 2:
Enter a source string:
abcdefg
Enter a target string:
abcde123
stringcmp(): greater than

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

Q4 – 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++;
   }
}
```



Comparison is based on ASCII value