Character Strings – Q1

Write two versions of a C function that remove all the blank spaces in a sentence. **The** first version **sweepSpace1()** will use array notation for processing the string, and the other version **sweepSpace2()** will use pointer notation. The function prototypes are given below:

```
char *sweepSpace1(char *sentence);
   // use array notation for accessing array elements
char *sweepSpace2(char *sentence);
   // use pointer notation for accessing array elements
```

Write a C program to test the function.

```
Enter a string:
i am a boy
sweepSpace1(): iamaboy
sweepSpace2(): iamaboy
```

Character Strings – Q1

```
#include <stdio.h>
#include <string.h>
char *sweepSpace1(char *sentence);
char *sweepSpace2(char *sentence);
int main()
{
   char str[80];

   printf("Enter the string: \n");
   gets(str);
   printf("sweepSpace1(): %s\n", sweepSpace1(str));
   printf("sweepSpace2(): %s\n", sweepSpace2(str));
   return 0;
}
```

Enter a string: *i am a boy*
sweepSpace1(): iamaboy
sweepSpace2(): iamaboy

Using array index for processing

```
char *sweepSpace1(char *sentence)  {
   int i, j, len;
   len = strlen(sentence);
   j = 0;
   for ( i=0; i < len; i++)
   {
      if (sentence[i] != ' ')
      {
         sentence[j] = sentence[i];
         j++;

      }
   }
   sentence[j] = '\0';
   return sentence;
}
```

Using pointer for processing

```
char *sweepSpace2(char *sentence)  {
   int i, j, len;
   len = strlen(sentence);
   j = 0;
   for ( i=0; i < len; i++)
   {
      if (*(sentence+i) != ' ')
      {
         *(sentence+j) = *(sentence+i);
         j++;
      }
   }
   *(sentence+j) = '\0';
   return sentence;
}
```

Character Strings – Q2

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:

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

In addition, write a C program to test the stringncpy function. Your program should read the string and the target *n* characters from the user and then call the function with the user input. In this program, you are not allowed to use any functions from the C standard String library.

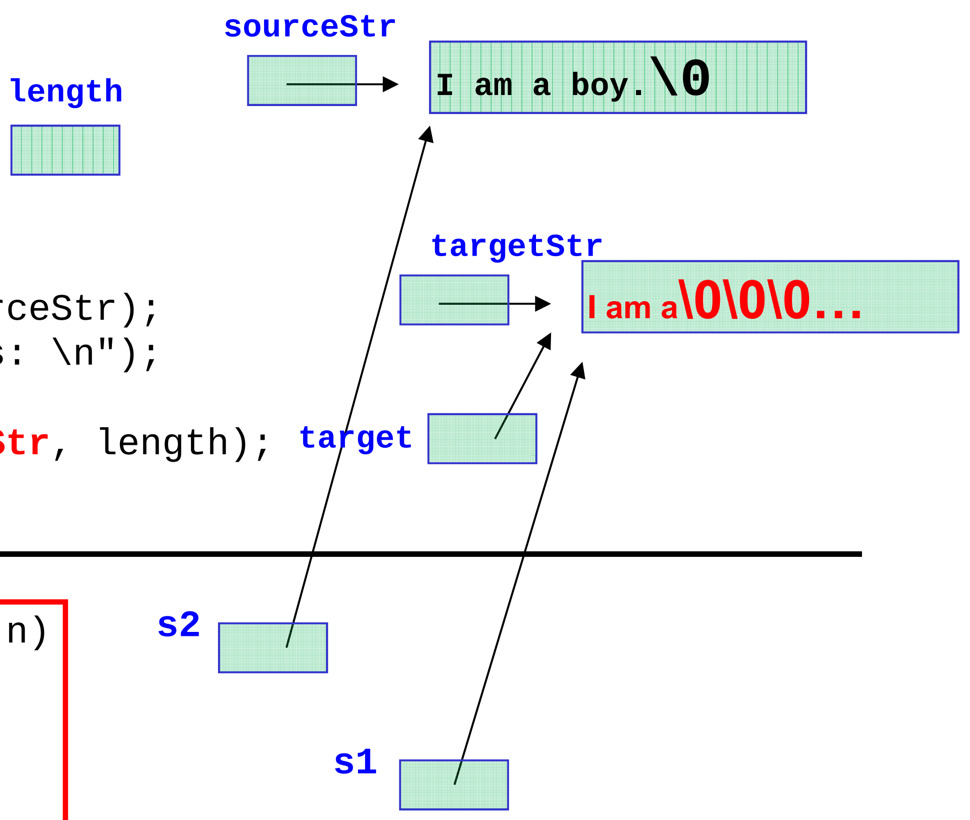
```
Enter a string:
I am a boy.
Enter the number of characters:
7
Returned string: I am a

Enter a string:
I am a boy.
Enter the number of characters:
21
Returned string: I am a boy.
```

Character Strings – Q2

```
#include <stdio.h>
char *stringncpy(char *s1, char *s2, int n);
int main()
{
   char sourceStr[40];
   char targetStr[40], *target;
   int length;
   printf("Enter the string: \n");
   gets(sourceStr);
   printf("The source string: %s\n", sourceStr);
   printf("Enter the number of characters: \n");
   scanf("%d", &length);
   target = stringncpy(targetStr, sourceStr, length);
   printf("stringncpy(): %s", 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;
}
```



Character Strings – Q3

Write a C program that reads and searches character strings. In the program, it contains a function **findTarget()** that searches whether a target name string has been stored in the array of strings. The function prototype is

int findTarget(char *target, char nameptr[SIZE][80], int size);

where *nameptr* is the array of strings entered by the user, *size* is the number of names stored in the array and *target* is the target string. If the target string is found, the function will return its index location, or -1 if otherwise.

```
Enter no. of names:
4
Enter 4 names:
Peter Paul John Mary
Enter target name:
John
findTarget(): 2
```

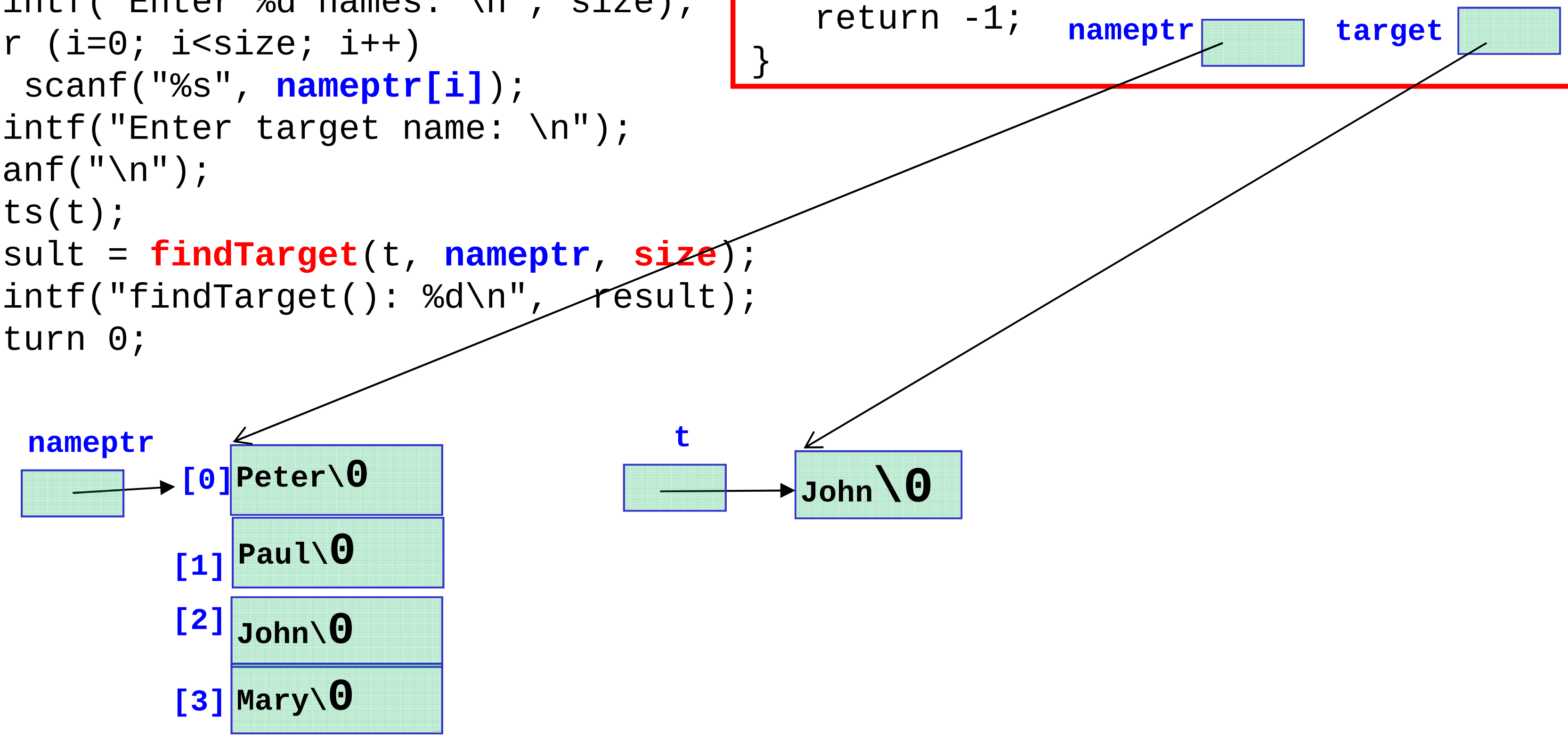
```
Enter no. of names:
5
Enter 5 names:
Peter Paul John Mary Vincent
Enter target name:
Jane
findTarget(): -1
```

Character Strings – Q3

```
#include <stdio.h>
#include <string.h>
#define SIZE 10
int findTarget(char *target, char
     nameptr[SIZE][80], int size);
int main(){
   char nameptr[SIZE][80];
   char t[40];
   int i, result, size;

   printf("Enter size: \n");
   scanf("%d", &size);
   printf("Enter %d names: \n", size);
   for (i=0; i<size; i++)
      scanf("%s", nameptr[i]);
   printf("Enter target name: \n");
   scanf("\n");
   gets(t);
   result = findTarget(t, nameptr, size);
   printf("findTarget(): %d\n",  result);
   return 0;
}
```

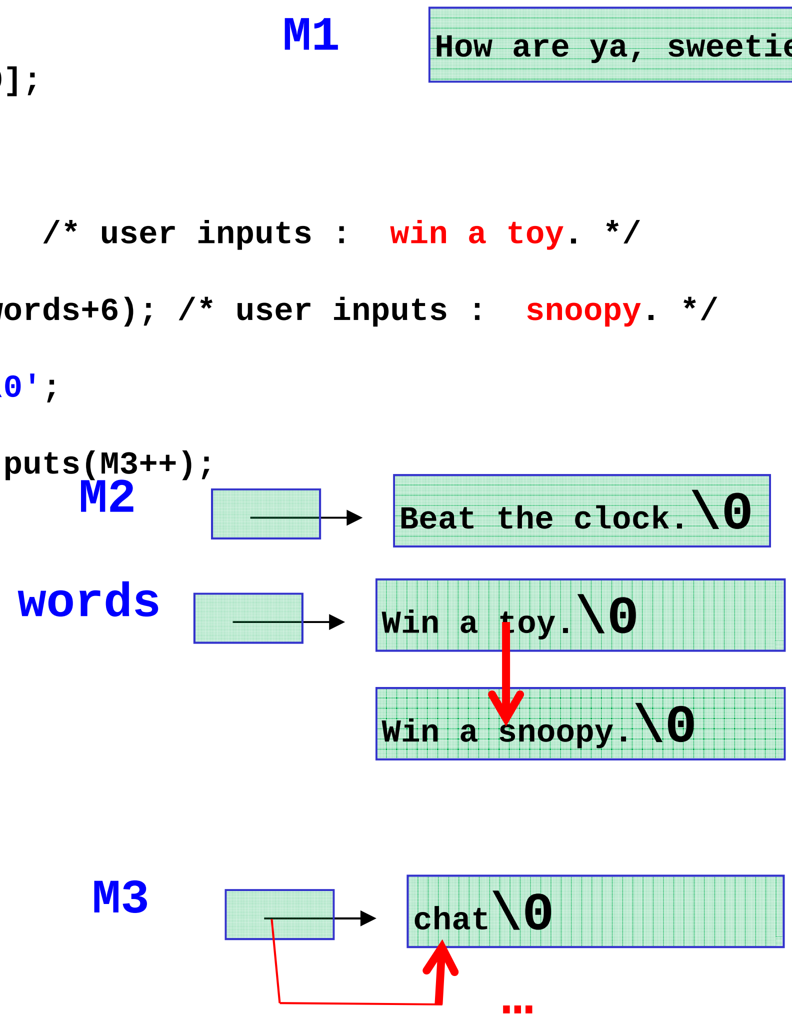
```
int findTarget(char *target, char
      nameptr[SIZE][80], int size)
{
   int i;
   for (i=0; i<size; i++) {
      if (strcmp(nameptr[i], target) == 0)
         return i;
   }
   return -1;
}
```



What is the output of the program?

Character Strings – Q4

```
#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;
}
```



```
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?
```