TIC1001—Introduction to Computing and Programming
National University of Singapore

Lecture 6: Strings and Arrays

Consider the following code

In [ ]:

```
long arr[10];
printf("%d %p %p", (int)sizeof(arr[0]),
                   &arr[0],
                   &arr[1]);
```

What are the possible outputs?

4 0x7fff42a0 0x7fff42a4
8 0x7fff42a0 0x7fff42a8

C specifications only say long has to be at least 4 bytes. Some systems might use more.

Uncle Soo thinks he does not need to explicitly pass in the size of the array into the function because he can use sizeof to find the size of the array. He wrote some code to test out his idea:

In [ ]:

```
void test(int arr[]) {
    printf("%lu", sizeof(arr));
}

int main(void) {
    int x[10] = {0};
    printf("%lu ", sizeof(x));
    test(x);
}
```

What is the output when Uncle Soo runs this code on his 64-bit OS? Try and see why your system shows.

40 8

Because arrays are passed by reference. So arr is a pointer which is either 64 bits (8 bytes) or 32 bits (4 bytes) depending on your system memory size.

Write a function sum_int which takes an array of integers and its size, and returns the sum of all its elements.

In [ ]:

```
int sum_int(int arr[], int size)
{
    int sum = 0;
    for (int i =0; i < size; i++)
    {
        sum += arr[i];
    }
    return sum;
}
```

Write a function max that takes in an array of int and its size, and returns the maximum value found in the array.

In [ ]:

```
int max(int arr[], int size)
{
    int max = arr[0];

    for (int i = 1; i < size; i++)
    {
        if (arr[i] > max)
        {
            max = arr[i];
        }
    }
    return max;
}
```

Write a function to_int which takes in a string of digits (the characters are all '0' - '9') and returns the number represented in the string as an int type.

Hint: What is the ASCII code for character '0'?

In [ ]:

```
#include <math.h>
int to_int(char str[])
{
    int toint=0, len= 0;

    for (int j=1; str[j] !='\0';j++)
    {
        len += 1;
        printf("len is: %d\n",len);
    }

    for (int i=0; str[i] !='\0';i++)
    {
        toint += (str[i] - 48) * pow(10,len-i);
        printf("toint is: %d\n",toint);
    }
    return toint;
}
```

Write a function to_upper that takes as input a string, and converts all its lowercase characters to uppercase.

Hint: The ASCII table might help.

In [1]:

```
void to_upper(char str[])
{
    for (int i = 0; str[i] !='\0';i++)
    {
       if(str[i] >= 'a' && str[i] <= 'z')
        {
            str[i] = str[i] - 32;
        }
    }
}
```

Write a function num_words that takes in a sentence as a string, and returns the number of words in the sentence.

You may assume that the words are separated by exactly one space and the sentence does not begin or end with a space.

Note: const char is required in the function parameter to allow us to pass string literals into the function.

In [ ]:

```
int num_words(const char str[])
{
    int len = strlen(str);
    int count = 0;
    for(int i = 0; i < len ; i++)
    {
        if (str[i] == 32)
        {
            count += 1;
        }
    }
    return count +1;
}
```

-END-