SC1003
Review Lecture
Week 11

Review Lecture – Week 11

- **Week 11 – Learning Materials**
  - Lectures
  - Lab and Tutorial
  - LAMS MCQ Questions
  - Coding Practice Questions
- Reviews on Character Strings
- Examples

Learning Schedule (Week 8 – Week 13)

Week	**Week 8** **4 Oct**	**Week 9** **11 Oct**	**Week 10** **18 Oct**	**Week 11** **25 Oct**	**Week 12** **1 Nov**	**Week 13** **8 Nov**	**Week 14** **15 Nov**
Topics	**Basic C Programming and Control Flow**	**Functions and Pointers**	**Arrays**	**Character Strings**	**Structures**		**Lab Test (MCQ Test & Coding Test)** **Dates: 15 Nov (Mon) and 16 Nov (Tue)** **Details will be announced when confirmed.**
Review Lecture	Date: **4 Oct 2021 (Monday)** Time: 9:30am-10:30am Online: **MS Teams** (the link for the online lecture is given at the end of the table)	Date: **11 Oct 2021 (Mon)** Time: 9:30am-10:30am Online: **MS Teams** (see below for the link for online lecture)	Date: **18 Oct 2021 (Mon)** Time: 9:30am-10:30am Online: **MS Teams** (see below for the link for online lecture)	Date: **25 Oct 2021 (Mon)** Time: 9:30am-10:30am Online: **MS Teams** (see below for the link for online lecture)	Date: **1 Nov 2021 (Mon)** Time: 9:30am-10:30am Online: **MS Teams** (see below for the link for online lecture)		
e-Learning Lectures	Learn: Course Introduction Learn: (1) Basic C Programming; (2) Control Flow	Learn: (1) Functions and (2) Pointers	Learn: (1) 1-D Arrays and (2) 2-D Arrays	Learn: Character Strings	Learn: Structures		
Lab-Tutorial	Learn: CodeBlocks IDE Do: Lab-Tutorial 1 (Qns are also available in APAS)	Do: Lab-Tutorial 2 (Qns are also available in APAS)	Do: Lab-Tutorial 3 (Qns are also available in APAS)	Do: Lab-Tutorial 4 (Qns are also available in APAS)	Do: Lab-Tutorial 5 (Qns are also available in APAS)		
Practice Questions	Learn: using APAS system Do: Coding Practice Questions (APAS>Quiz) Do: MCQ Questions (LAMS)	Do: Coding Practice Questions (APAS>Quiz) Do: MCQ Questions (LAMS)	Do: Coding Practice Questions (APAS>Quiz) Do: MCQ Questions (LAMS)	Do: Coding Practice Questions (APAS>Quiz) Do: MCQ Questions (LAMS)	Do: Coding Practice Questions (APAS>Quiz) Do: MCQ Questions (LAMS)		
Assignment	Learn: (1) Assignment Submission and Grading process; (2) Review Request Form (Procedure)		**Assignment paper – Available in APAS**			**Assignment due**	

Lecture Video – Character Strings

- **Watch Lecture Video – Character Strings**
  (NTULearn: C Programming > E-Learning Lectures > Week 11)

6 - Character Strings - Lecture

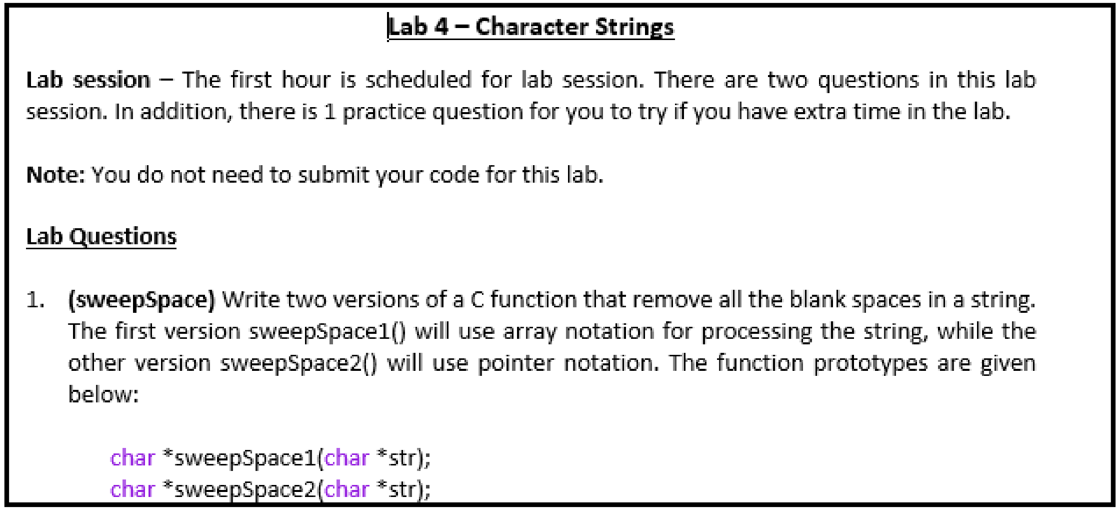
6

Character Strings

Week 11 - Lab

- **Lab 4 – Character Strings**

(NTULearn: C Programming > Lab-Tutorials > Lab-Tutorial 4)



Lab 4 – Character Strings

Lab session – The first hour is scheduled for lab session. There are two questions in this lab session. In addition, there is 1 practice question for you to try if you have extra time in the lab.

Note: You do not need to submit your code for this lab.

Lab Questions

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

```
char *sweepSpace1(char *str);
char *sweepSpace2(char *str);
```

Suggested solutions:
Available in the same folder. You may refer to the suggested code if you have any difficulty in attempting the lab questions.

Lab Coding Questions:

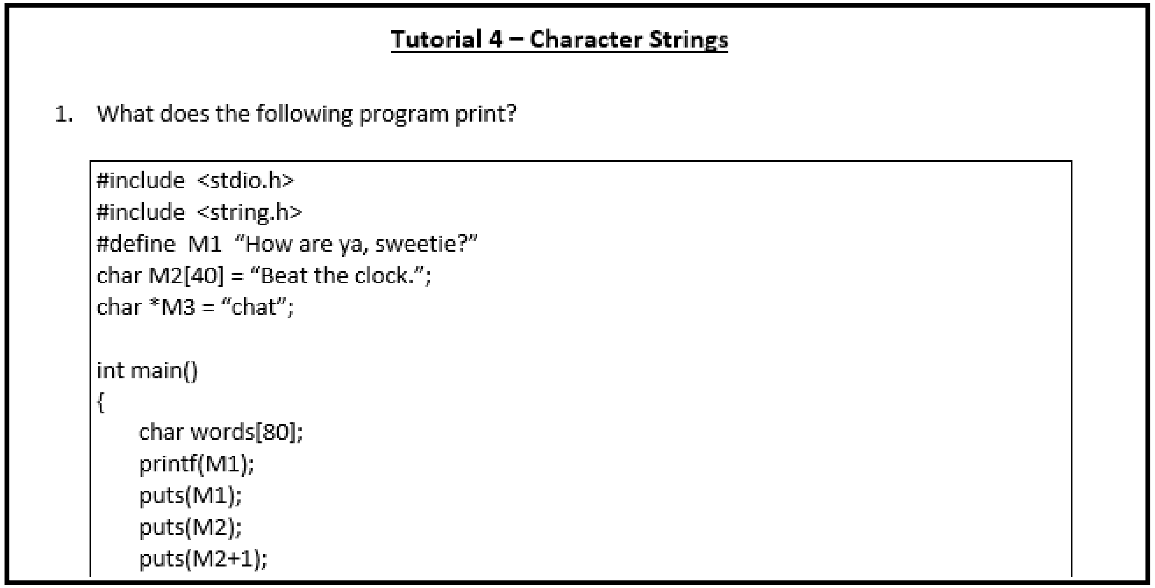
- sweepSpace
- findTarget
- palindrome

Available at: APAS > Exercise (choose Topic): You may test your code with sample test cases in APAS.

Week 11 - Tutorial

- **Tutorial 4 – Character Strings**

(NTULearn: C Programming > Lab-Tutorials > Lab-Tutorial 4)



Tutorial 4 – Character Strings

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];
      printf(M1);
      puts(M1);
      puts(M2);
      puts(M2+1);
```

Tutorial Coding Questions:

- stringncpy
- stringcmp

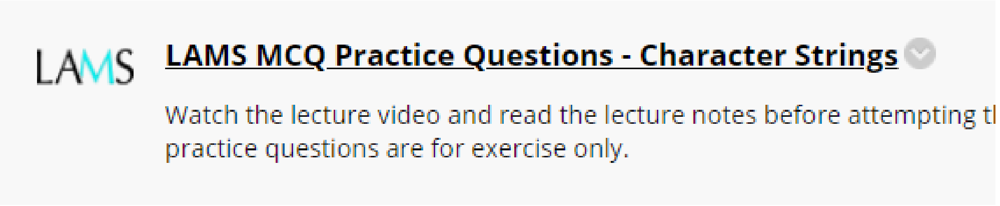
Suggested solutions: Available at the end of each week in the same folder in NTULearn.

Available at: APAS > Exercise (choose Topic): You may test your code with sample test cases in APAS.

LAMS MCQ Questions

- **LAMS MCQ Questions – Character Strings**

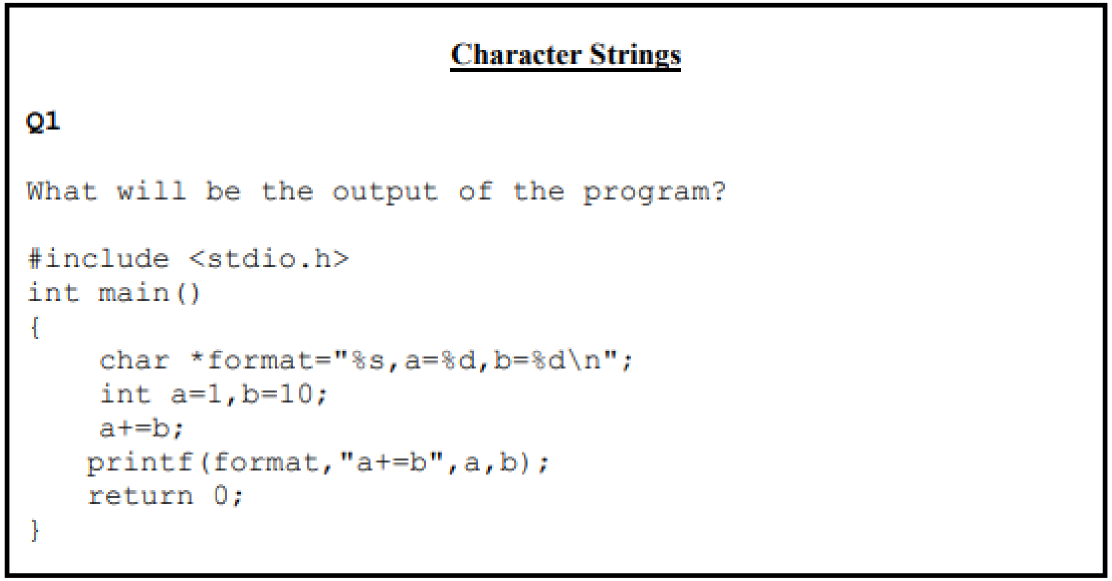
  (NTULearn: C Programming > LAMS MCQ Questions > Character Strings)



LAMS

LAMS MCQ Practice Questions - Character Strings

Watch the lecture video and read the lecture notes before attempting tl practice questions are for exercise only.



Character Strings

Q1

What will be the output of the program?

```
#include <stdio.h>
int main()
{
    char *format="%s,a=%d,b=%d\n";
    int a=1,b=10;
    a+=b;
   printf(format,"a+=b",a,b);
   return 0;
}
```

Answers and explanations on each question are available in the same folder.

APAS - Coding Practice Questions

- **Coding Practice Questions – Character Strings**

  (APAS: Quiz > Character Strings)

1. insertChar
2. locateFirstChar
3. stringrChr
4. processString
5. longWordLength
6. countWords
7. cipherText
8. longestStrInAr
9. findMinMaxStr
10. maxCharToFront
11. findSubstring

Suggested solution can be found at (need VPN is accessing from outside NTU):

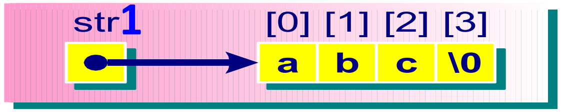
http://172.21.147.174/ > NTUQA

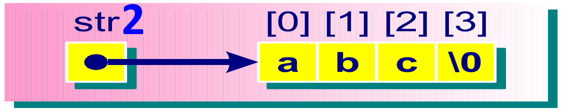
Review Lecture – Week 11

- Week 11 – Learning Materials
  - Lectures
  - Lab and Tutorial
  - LAMS MCQ Questions
  - Coding Practice Questions
- **Reviews on Character Strings**
- Examples

Python vs C – Declaring Character Strings

	Python	**C**

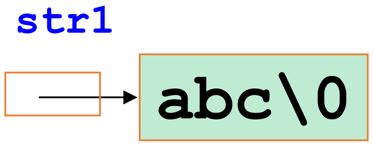
string constant and initialization	string = “abc”
	[immutable]
		• A string is an array of characters terminated by a **null character** (‘\n’)
		// using array notation – pointer constant
		char str1[] = “abc”;
		str1 == &str1[0]
		*str1 == str1[0] == ‘a’
		*(str1+1) == str1[1] == ‘b’
		++str1 // illegal
		str1 = str2; // illegal
		

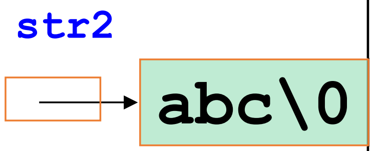
string variable and initialization	string = ‘’	**// using pointer notation – pointer variable**
		char *str2 = “abc”;
		str2 == &str2[0]
		*str2 == str2[0] == ‘a’
		++str2; // ok
		str2=str1; // ok
		

string element assignment (indexing}	string[2] = ‘d’;	str1[2] = ‘d’;

Python vs C – String Processing (Example)

	Python	C

iterating over strings using indexing	string = “Python Programming”
	for char **in** string:
	  if (string[char] == ‘a’):
	    count += 1
	print(“count = ”, count);
		char str1[] = “abc”; // pointer constant
		int count=0;
		int i;
		while (str1[i] != **’\0’**) { **// using index**
		  if (str1[i] == ‘a’)
		    count++;
		  i++;
		}
		printf(“count = “%d”, count);
		
		
		output
		count = 1

iterating over arrays using pointers	No real equivalent in Python.	**char *str2 = “abc”; // pointer variable**
		int count=0;
		while (*str2 != ’\0’) { **// using pointer**
		  if (*str2 == ‘a’)
		    count++;
		  str2++;
		}
		printf(“count = “%d”, count);
		
		
		output
		count = 1

Python vs C – String Functions/Methods

Python

Operations

+ - concatenate strings

* - repeat string

slicing: string[start:finish:step]

String functions

len(), chr(), ord(), input(), etc.

String methods

upper(), find(), index(), join(), lower(), replace(), split(), upper(), format(), etc.

C

String Input/Output

[#include <stdio.h>] fgets() [instead of gets()], puts(), scanf(), printf(), etc.

- **Note the difference between scanf() and fgets().**
- We use **fgets()** instead of **gets()** because **gets()** is not safe as it does not check the array bound.

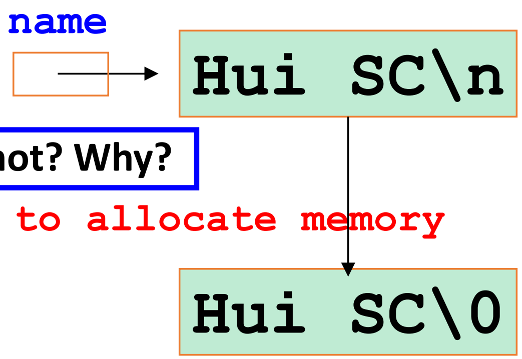
String Functions

[#include <string.h>] strrchr(), **strcat()**, strncat(), strchr(), **strcmp()**, strncmp(), **strcpy()**, strncpy(), strpbrk(), **strlen()**, etc.

Example:

```
#include <stdio.h>
#include <string.h>
int main() {
   char name[80],*p; // MUST use array to allocate memory
   printf("Hi, what is your name?\n");
   fgets(name, 80, stdin);
   if (p=strchr(name,'\n')) *p = '\0';
   printf("Nice name, %s.\n", name);
   return 0;
}
```

Q: char *name; Ok or not? Why?



Python vs C – String Functions/Methods

Python

Example:

```python
student_one = input("Ename name one: ")
student_two = input("Enter name two: ")

if student_one < student_two:
   print(student_one + " comes before " +
student_two + " in the alphabet.")
elif student_one > student_two:
   print(student_two + " comes before " +
student_one + " in the alphabet.")
else:
   print("They are the same in the alphabet.")
```

Program Input and Output

```
Enter name one: John
Enter name two: Mary
John comes before Mary in
the alphabet.
```

C

Example on using strcmp():

```c
#include <stdio.h>
#include <string.h>
int main()
{
  char student_one[80], student_two[80];   // note: must use array notation here

  printf("Enter name one: ");
  scanf("%s", student_one);
  printf("Enter name two: ");
  scanf("%s", student_two);

  if (strcmp(student_one, student_two) < 0)
    printf("%s comes before %s in the alphabet.\n", student_one, student_two);
  else if (strcmp(student_one, student_two)<0)
    printf("%s comes before %s in the alphabet.\n", student_two, student_one);
  else
    printf("They are the same in the alphabet.\n");
  return 0;
}
```

When comparing two strings, must use the strcmp() function, do not use relational operators (e.g. ==, >=, etc.) ; similarly for strcpy().

Python vs C – Character Functions/Methods

Python

Character Methods: isalnum(), isalpha(), isdecimal(), isdigit(), islower(), isnumeric(), isprintable(), isspace(), isupper(), etc.

Example:

```python
def convert(string):
  newString = ''
  for char in string:
    if (char.isupper() == True):
      newString += char.lower()
    elif (char.islower() == True):
      newString += char.upper()
    else:
      newString += char
  return newString
def main():
  newString = convert("Python Programming")
  print(newString)
main()
```

C

Character Functions: **[#include <ctype.h>]** isalnum(), isalpha(), isdigit(), islower(), ispunct(), isupper(), isspace(), isxdigt(), toupper(), tolower(), etc.

- These functions are used to test the nature of a character. Return **true (non-zero)** if the character belongs to a particular class, and return **false (zero)** otherwise.

Example:

```c
#include <stdio.h>
#include <ctype.h>
void convert(char *);
int main(){
  char str[80]=“This is a test";
  convert(str); puts(str);
  return 0;
}
void convert(char *s){
  while (*s != '\0') {
    if (isupper(*s))   *s = tolower(*s);
    else if (islower(*s))   *s = toupper(*s);
    s++;
  }
}
```

Program Output

```
This is a test
tHIS IS A TEST
```

Python vs C – String to Number Conversions

Python

Example:

```python
num1=int("10")
num2=float("12.34")
print("{:d} {:.2f}".format(num1,num2))
```

C

String to Number Functions:

[#include <stdlib.h>] atof(), atoi()

- There are two ways to store a number. It can be stored as strings or in numeric form. Sometimes, it is convenient to read in the numerical data as a string and convert it into the numeric form. To do this, C provides the functions: **atoi()** and **atof()**.

Example:

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
  int num1;
  double num2;
  char ar1[10]="10";
  char ar2[10]="10.2";
  num1 = atoi(ar1);
  num2 = atof(ar2);
  printf("%d %.2f",num1,num2);
  return 0;
}
```

Program Output

```
10 10.20
```

Python vs C – Formatted String I/O

Python

May use the string method split().

Example:

```
str1 = "string 1369 531"
str2,b,c = str1.split()
i = int(b)
j = int(c)
str3 = str(j)+" "+ str(i) + " " + str2
print(str3)
```

Program Output

```
531 1369 string
```

C

Formatted String I/O Functions

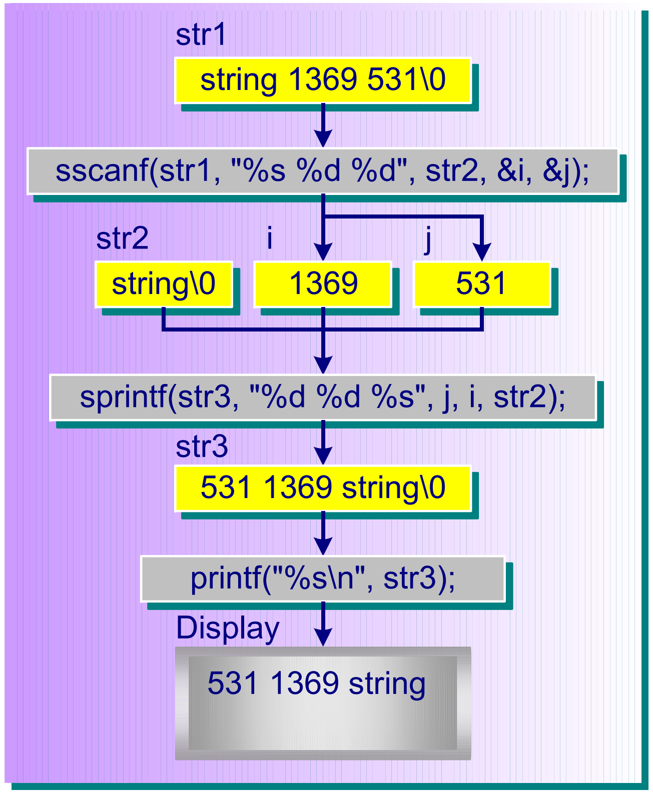
[#include <stdio.h>] sscanf(), sprintf()

Example:

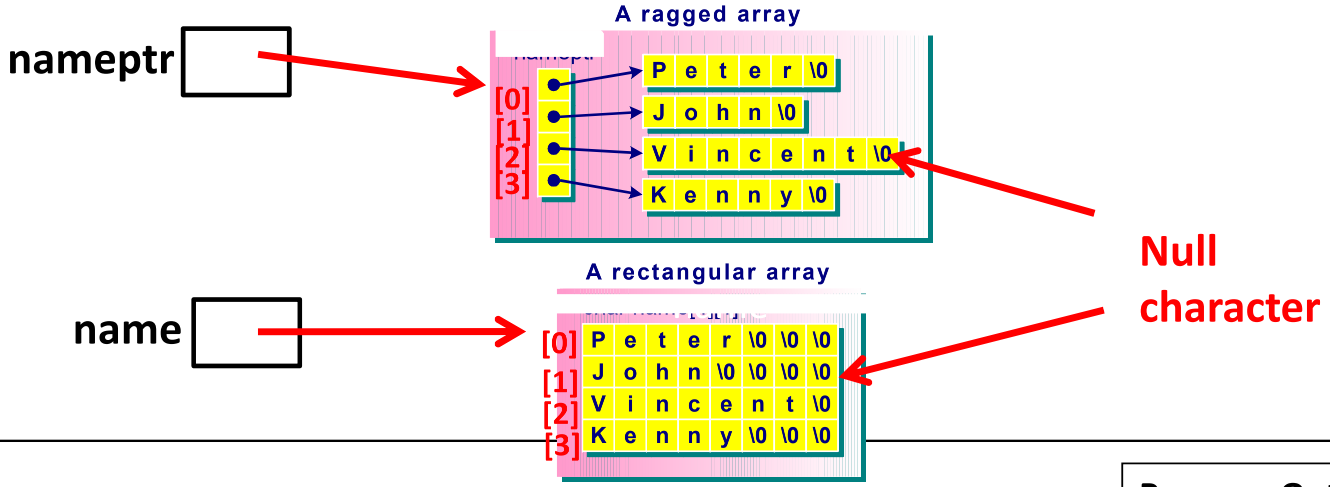
```
#include <stdio.h>
int main()
{
  char  str1[80] = "string 1369 531";
  char  str2[80], str3[MAX_CHAR];
  int   i, j;

  sscanf(str1, "%s %d %d", str2, &i, &j);
  sprintf(str3, "%d %d %s", j, i, str2);

  printf("%s\n", str3);
  return 0;
}
```



C - Arrays of Character Strings



```c
#include <stdio.h>
int main(){
   char *nameptr[4] = {"Peter","John","Vincent","Kenny"};
   char name[4][8] = {"Peter","John","Vincent","Kenny"};
   int  i,j;
   printf("Ragged Array: \n");
   for (i=0; i<4; i++)
      printf("nameptr[%d] = %s\n", i, nameptr[i]);
   printf("Rectangular Array: \n");
   for (j=0; j<4; j++)
      printf("name[%d] = %s\n", j, name[j]);
   return 0;
}
```

Program Output

```
Ragged Array:
nameptr[0] = Peter
nameptr[1] = John
nameptr[2] = Vincent
nameptr[3] = Kenny
Rectangular Array:
name[0] = Peter
name[1] = John
name[2] = Vincent
name[3] = Kenny
```

Review Lecture – Week 11

- Week 11 – Learning Materials
  - Lectures
  - Lab and Tutorial
  - LAMS MCQ Questions
  - Coding Practice Questions
- Reviews on Character Strings
- **Examples**

Example 1 – countStrings

The following program calculates the number of input strings with letter ‘a’, and end the program when the input string is “####”.

Program Input and Output:

```
enter a string (enter #### to stop): apple
enter a string (enter #### to stop): banana
enter a string (enter #### to stop): strawberry
enter a string (enter #### to stop): book
enter a string (enter #### to stop): ####
3 strings with letter 'a'
```

Example 1 – Suggested Code

Python:

```python
count = 0
str_sentinal = input("enter a string (enter #### to stop): ")
while str_sentinal != "####":
  for letter in str_sentinal:
    if letter == 'a':
      count +=1
      break
  str_sentinal = input("enter a string (enter #### to stop): ")
print(count , "strings with letter 'a'")
```

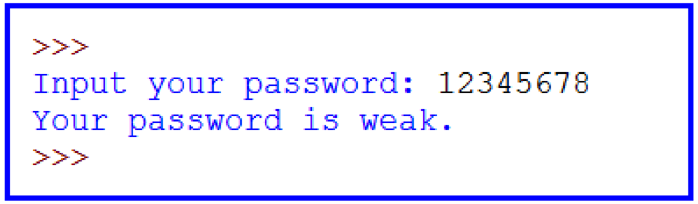
C:

```c
#include <stdio.h>
#include <string.h>
int main() {
  int count=0, i=0;
  char str_sentinal[20];
  printf("enter a string (enter #### to stop): ");
  scanf("%s", str_sentinal);
  while (strcmp(str_sentinal,"####")!=0) {
    while (str_sentinal[i] != '\0') {
      if (str_sentinal[i] == 'a') {
        count+=1;
        break;
      }
      i++;
    }
    printf("enter a string (enter #### to stop): ");
    scanf("%s", str_sentinal);
  }
  printf("%d strings with letter 'a'", count);
  return 0;
}
```

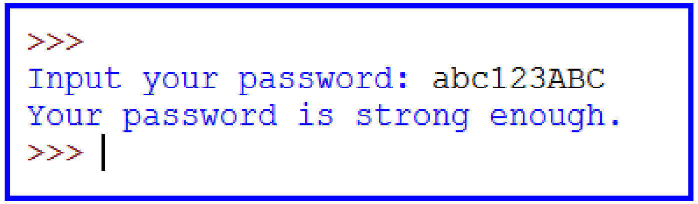
Example 2 – password

- When choosing a password for online accounts, there are typically certain requirements for the strength of the password.
- Develop a Python program for testing if a string satisfies some appropriate criteria for a strong password.
- It’s up to you to define the requirements.

Program Input and Output:



```
>>>
Input your password: 12345678
Your password is weak.
>>>
```



```
>>>
Input your password: abc123ABC
Your password is strong enough.
>>> |
```

Example 2 – Suggested Code

Python:

```python
LENGTH = 8
password = input("Input your password: ")
upCase = False
lowCase = False
digit = False
for char in password:
  if char.isupper():
    upCase = True
  if char.islower():
    lowCase = True
  if char.isdigit():
    digit = True
length = len(password)
strong = upCase and lowCase and digit and length>LENGTH
if strong:
  print("Your password is strong enough.")
else:
  print("Your password is weak.")
```

C:

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define LENGTH 8
int main() {
  char password[20]; int upCase=0, lowCase=0, digit=0;
  int i=0, strong, length;
  printf("Input your password: ");
  scanf("%s", password);
  while (password[i] != '\0') {
    if (isupper(password[i]))
      upCase = 1;
    if (islower(password[i]))
      lowCase = 1;
    if (isdigit(password[i]))
      digit = 1;
    i++;
  }
  length = strlen(password);
  strong = upCase && lowCase && digit && (length > LENGTH);
  if (strong == 1)
    printf("Your password is strong enough.");
  else
    printf("Your password is weak.");
  return 0;
}
```

Example 3 – longWordLength

Write a C function that accepts an English sentence as parameter, and returns the length of the longest word in the sentence. For example, if the sentence is "I am happy.", then the length of the longest word "happy" in the sentence 5 will be returned. Assume that each word is a sequence of English letters.
The function prototype is given as follows:

```
int longWordLength(char *s);
```

Program input and output

```
Enter a string:
I am happy.
longWordLength(): 5
```

Example 3 – Suggested Code

Python:

```
def longWordLength(aString):
  strLst = aString.split()
  max = len(strLst[0])
  for i in range(1,len(strLst)):
    if (len(strLst[i]) > max):
      max = len(strLst[i])
  return max

def main():
  inputString = input("Enter a string: ")
  max = longWordLength(inputString)
  print(max)

main()
```

C:

```
#include <stdio.h>
#include <string.h>
int longWordLength(char *s);
int main() {
  char str[80], *p;
  printf("Enter a string: \n");
  fgets(str, 80, stdin);
  if (p=strchr(str,'\n')) *p = '\0';
  printf("longWordLength(): %d\n", longWordLength(str));
  return 0;
}
int longWordLength(char *s){
  int max=0,len=0;
  while ( *s!='\0' ) {
    while ( ( (*s<='Z') && (*s >='A’) ) || ( (*s<='z') && (*s>='a’) ) )  {
      len++; s++;
    }
    if (len>max) max=len;
    len=0; s++;
  }
  return max;
}
```

Example 4 – maxCharToFront()

Write a C function maxCharToFront() that accepts a character string *str* as parameter, finds the largest character from the string (based on ASCII value), and moves it to the beginning of the string. E.g., if the string is "adecb", then the string will be "eadcb" after executing the function. The string will be passed to the caller via call by reference. If more than one largest character is in the string, then the **first appearance** of the largest character will be moved to the beginning of the string. For example, if the string is "adecbe", then the resultant string will be "eadcbe".

The function prototype is given as follows:

```
void maxCharToFront(char *str);
```

Program input and output:

```
Enter a string:
adebc
maxCharToFront(): eadbc

Enter a string:
afgcdeg
maxCharToFront(): gafcdeg
```

Example 4 – Suggested Code

Python: main

```python
def main():
  aString = input("Enter a string: ")
  newString = maxcharToFront(aString)
  print("maxCharToFront(): ",newString)

main()
```

C: main()

```c
#include <stdio.h>
#include <string.h>
void maxCharToFront(char *str);
int main()
{
  char str[80], *p;

  printf("Enter a string: \n");
  fgets(str, 80, stdin);
  if (p=strchr(str,'\n')) *p = '\0';
  printf("maxCharToFront(): ");
  maxCharToFront(str);
  puts(str);
  return 0;
}
```

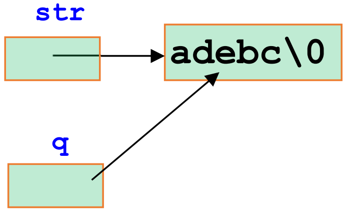
Example 4 – Suggested Code

Python:

```python
def maxcharToFront(aString):
    newString = aString[:]
    max = newString[0]
    index = 0
    for i in range(1, len(newString)):
        if (max < newString[i]):
            max = newString[i]
            index = i
    aString = '' + max
    aString += newString[0:index]
    for i in range(index+1, len(newString)):
        aString += newString[i]
    return aString
```

C:

```c
void maxCharToFront(char *str) {
    char max,*q;
    int i=0;
    max=str[0];
    q=str;
    while (str[i] != '\0') {
        if (max<str[i]) {
            max=str[i];
            q=str+i;
        }
        i++;
    }
    while (q>str) {
        *q=*(q-1);
        q--;
    }
    str[0]=max;
}
```



Example 5 – cipherText

Cipher text is a popular encryption technique. What we do in cipher text is that we can encrypt each alpha ('a' .. 'z', 'A' .. 'Z') character with +1. For example, "Hello" can be encrypted with +1 cipher to "Ifmmp". If a character is 'z' or 'Z', the corresponding encrypted character will be 'a' or 'A' respectively. For other characters, no encryption is performed. We use call by reference in the implementation.

Write the C functions cipher() and decipher() with the following function prototypes:

```c
void cipher(char *s);
void decipher(char *s);
```

Program input and output

```
Enter a cipher string:
123a
cipher(): 123b
decipher(): 123a

Enter a cipher string:
HELLO Hello
cipher(): IFMMP Ifmmp
decipher(): HELLO Hello
```

Example 5 – Suggested Code

Python: main

```python
def main():
   inputString = input("Enter a cipher text: \n")
   aString = cipher(inputString)
   print("cipher(): ",aString)
   dString = decipher(aString)
   print("decipher(): ",dString)

main()
```

C: main

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
void cipher(char *s);
void decipher(char *s);
int main()
{
   char str[80], *p;

   printf("Enter a cipher string: \n");
   fgets(str, 80, stdin);
   if (p=strchr(str,'\n')) *p = '\0';
   cipher(str);
   printf("cipher(): %s\n", str);
   decipher(str);
   printf("decipher(): %s", str);
   return 0;
}
```

Example 5 – Suggested Code

Python:

```python
def cipher(aString):
  newString = aString[:]
  aString = ''
  for char in newString:
    if (char.isalpha() == True):
      if (char == 'z'):
        aString += 'a'
      elif (char == 'Z'):
        aString += 'A'
      else:
        aString += chr(ord(char)+1)
  return aString
```

```python
def decipher(aString):
  newString = aString[:]
  aString = ''
  for char in newString:
    if (char.isalpha() == True):
      if (char == 'a'):
        aString += 'z'
      elif (char == 'A'):
        aString += 'Z'
      else:
        aString += chr(ord(char)-1)
  return aString
```

C:

```c
void cipher(char *s)
{
  char *str;
  int i, len;

  len = strlen(s);
  for (i=0; i<len; i++){
    if (isalpha(s[i])) {
      if (s[i] == 'z')
        s[i]='a';
      else if (s[i] == 'Z')
        s[i]='A';
      else
        s[i]=s[i] + 1;
    }
  }
}
```

```c
void decipher(char *s)
{
  char *str;
  int i, len;

  len = strlen(s);
  for (i=0; i<len; i++){
    if (isalpha(s[i])) {
      if (s[i] == 'a')
        s[i]='z';
      else if (s[i] == 'A')
        s[i]='Z';
      else
        s[i]=s[i] - 1;
    }
  }
}
```

Thank you !!!