COMP10002 Workshop Week 7

1	char and strings
2	discuss 7.12, 7.14, 7.15, 7.16
LAB	Implement at least two from 7.12 7.14 7.15, 7.16 OR Assignment 1
ASS1	Progress so far? Q&A
	Workshops: Discuss the requirements of Exercises 7.12 (palindromes), 7.14 (atoi), 7.15 (anagrams), and 7.16 (frequency counting for words); then Implement solutions to at least two of them (for some of them you will need to write simple main programs in order to test your functions; for some of them you can start with the insertionsort.c program and then alter it); plus There are also exercises in the lec06.pdf slides that you shouldn't ignore. **In the second half of the workshop, work on Assignment 1.**

Notes from lecture: lec06.pdf

introduce ctype.h & string.h

I string1.c - string vs pointer value

I strcpy.c - array and pointer notation

I getword.c - word= sequence of isalpha

- skipping non-alpha, getting all alpha to word

I words.c - also typedef char word_t[MAXCHARS+1];

I progargs.c - demon argc, argv

char and <ctype.h>

```
int c; while ( (c= getchar())!=EOF ) { /*proccess c */ }
char c; while ( (scanf("%c",&c)==1 )  { /*proccess c */ }
```

How to check if `c` is:

- a lower-case letter?

```
if (?) printf ("a lower-case letter!\n";
```

- an upper-case letter?

```
if ( ?) ...
```

- or just a letter?

```
if ( ? ) ...
if ( ? ) ...
```

In doubt on how to use a function, say `sqrt`?
Google it, or in Terminal type "`man sqrt`".

char and <ctype.h>

```
int c;
c= getchar();
```

How to check if `c` is:

- a digit?

```
if ( ? )
    printf ("a digit!\n");
if ( ? ) ...
```

- a letter or digit

```
if ( ? ) ...
```

- a white space such as space, \n, \t, \r …

```
if ( ? ) ...
```

- a punctuation such as . ,

```
if ( ? ) ...
```

check if c is :	not using library functions	using functions in ctype.h
- a digit	`c>='0' && c<='9'`	`isdigit(c)`
- a lower-case letter	`c>='a' && c<= 'z'`	`islower(c)`
- a letter	`(c>='a' && c<='z') || (c>='A' && c<='Z')`	`isalpha(c)`
- a white space such as `' '`, `'\n'`, `'\r'`, …	?	`isspace(c)`
- a punctuation	?	`ispunct(c)`

There are many other useful functions in ctype.h (google it), such as `toupper`, `tolower`, ….

arrays of char and strings

```
#define MAX_N 5
```

```
char A[MAX_N];
int n= 0;

A[0]= '1';
A[1]= '2';
A[2]= '3';
n= 3;
```

arrays of char and strings

```
#define MAX_N 5
```

```
char A[MAX_N];
int n= 0;

A[0]= '1';
A[1]= '2';
A[2]= '3';
n= 3;
```

`A` is an array of 5 characters.
`A` could be considered as a string if used appropriately (ie. with the terminating '`\0`').

```
char s[MAX_N + 1];


s[0]= '1';
s[1]= '2';
s[2]= '3';
s[3]= '\0';
```

`s` is an array of 6 characters
`s` is considered as a string, that can hold a string of up to 5 characters. Note that the string (ie. the content of the array) can change, but `s` itself is a *pointer constant* and can not be changed.

array notation & string notation for char *

Here, **s** is a variable of type **char *** , and it is initialised with the pointer to *constant string* **"12345"**.

```
char *s = "12345";
int n= strlen(s);
int i;

for (i=0; i<n; i++) {
  printf("%c", s[i]);
  //wrong: s[0]= 'A';
}
```

Array Notation

```
char *s = "12345";

char *p;

for (p=s; *p; p++){
  printf("%c", *p);
}
// note: here we cannot chage s[i] because the memory "12345" is a const; We can, however, change s by making s points to a different string
```

Pointer Notation

Examples: Array and String Notations

1. Write a function that returns the length of a string.
2. Write a function that returns number of alphabetic characters in a string.

Examples: Array and String Notations

1. Write a function that returns the length of a string.

```
int my_strlen(char *s) {
   int len;
   // Using pointer notation
   char *p;
   for (len=0, p=s; *p; len++, p++);
   //OR using array notation
   for (len=0; s[len]; len++);
   return len;
}
```

2. Write a function that returns number of alphabetic characters in a string.

Strings and <string.h>

```
char *s="1234";
char A[100]="abc";
```

Which of the following statements are OK:

```
1. *s = "567";
2. s = '5';
3. s = "56789012";
4. A = "defghijk";
5. printf("%d\n", strlen(s+1));
6. strcpy(A, "def");
7. strcpy(s, "012");
8. s[9]= '\0';
```

Strings and <string.h>

```
char *s="1234";        // s is a variable, "1234" is a
constant string
char A[100]="abc";     // constant "abc" is just used
for initialising array A[]
```

Which of the following statements are OK:

```
1.  *s = "567";          // error: type mismatch
2.  s = '5';             // error: type mismatch
3.  s = "56789012";      // ok, s points to new constant
4.  A = "defghijk";      // wrong, A is a const
5.  printf("%d\n", strlen(s+1)); // ok
6.  strcpy(A, "def"); // ok
7.  strcpy(s, "012"); // not ok: attempt to overwrite a
    constant string area
8.  s[9]= '\0';          // not ok, just like the above
```

Ex 7.12 (palindrome)

Write a function

```
int is_palindrome(char *)
```

that returns true if its argument string is a *palindrome*, that is, reads exactly the same forwards as well as backwards; and false if it is not a palindrome.

For example, `"rats live on no evil star"` is a palindrome according to this definition, while "`A man, a plan, a canal, Panama!`" is not. (But note that the second one is a palindrome according to a broader definition that allows for case, whitespace characters, and punctuation characters to vary.)

See `palindrome.net` for some interesting palindromes.

Preparation: Download `string_examples_skel.c` from `github.com/anhvir/c102`

Extra work: Change the function to satisfy the broader definition.

Ex 7.12: Palindrome

Ex 7.14: atoi

Write a function

```
int atoi(char *)
```

that converts a character string into an integer value.

Strings

```
char *s="123";
int n;
```

Which of the following fragments give the same result as `n=atoi(s)`:

```
1. for (; isdigit(*s); s++)
       n= n*10 + (*s);
2. for (n=0; isdigit(*s); s++)
       n= n*10 + (*s)
3. for (n=0; *s && isdigit(*s); s++)
       n= n*10 + (*s);
4. none of the above
```

Strings

```
char *s="123";
int n;
```

Which of the following fragments give the same result as `n=atoi(s)`:

```
1.  for (; isdigit(*s); s++)
      n= n*10 + (*s);
2. for (n=0; isdigit(*s); s++)
      n= n*10 + (*s)
3. for (n=0; *s && isdigit(*s); s++)
      n= n*10 + (*s);
4. none of the above  ➔ correct answer
```

The right segment for atoi can be obtained from 2:

```
for (n=0; isdigit(*s); s++)
      n= n*10 + (*s-'0');
```

Labs:

1. Implement **Ex 7.12** (palindrome), **7.14** (atoi), **7.15** (anagram), **7.16**

Labs:

1. Implement **Ex 7.12** (palindrome), **7.14** (atoi),
2. **7.15** (anagram, same as lec06.E3), **7.16**.

3. Suppose that a *word* is a sequence of maximal 20 alphabetic characters.
   - Declare a data type `word_t` accordingly.
   - Write a function `int first_word(char *s, word_t first)` that:
     - makes `first` be the first word that appears in string `s` if such exists, and returns true;
     - returns false if otherwise, and in this case `first` should not be modified.

4. Write a program that reads in a text and prints out the first word.

ASS2 Q&A

Labs? Other exercises from lec06.pdf

Exercise 1 Write a function `is_subsequence(char *s1, char *s2)` that returns 1 if the characters in `s1` appear within `s2` in the same order as they appear in `s1`. For example, `is_subsequence("bee", "abbreviate")` should be 1, whereas `is_subsequence("bee", "acerbate")` should be 0.

Exercise 2 Ditto arguments, but determining whether every occurrence of a character in `s1` also appears in `s2`, and 0 otherwise. For example, ", `is_subset("bee "rebel")` should be 1, whereas `is_subset("bee", "brake")` should be 0.

Exercise 3 Write a function `is_anagram(char *s1, char *s2`) that returns 1 if the two strings contain the same letters, possibly in a different order, and 0 otherwise, ignoring whitespace characters, and ignoring case. For example, `is_anagram("Algorithms", "Glamor Hits")` should return 1.

Exercise 4 Write a function `next_perm(char *s)` that rearranges the characters in a string argument and generates the lexicographically next permutation of the same letters. For example, if the string `s` is initially `"51432"`, then when the function returns s should be `"52134"`.

Exercise 5 If the two strings are of length `n` (and, if there are two, `m`), what is the asymptotic performance of your answers to Exercises 1–4?

NOT for Monday classes

Case Study & Ex 7.16 – The Task

Design and implement a program that reads text from stdin, and writes a list of the distinct words that appear, together with their frequencies.

First step:

Make sure you understand the task, that you can imagine what's the input and output.

Case Study & Ex 7.16 – Understanding The Task

Design and implement a program that reads text from stdin, and writes a list of the distinct words that appear, together with their frequencies.

Sample texts:

```
A cat in a hat!
+-abc 10e12 e 1abc #e#abc.abcdefghijklm=xyz
```

Output= ?

Assumptions/limits:

- Word=
- ? what else?
- ?

Case Study & Ex 7.16 - Design

What's input? What's the best (or just feasible) way to get it?

Case Study & Ex 7.16 - Design

How to store output, which data structure?

And how to produce output?

Case Study & Ex 7.16 – Alistair's getword

```
int getword(char W[], int limit) {
  int c, len=0;
  /* first, skip over any non alphabetics */
  while ((c=getchar()) != EOF && !isalpha(c)) {
      /* 12+34 aWord is the first word */
  }
  if (c==EOF)  return EOF;

  /* ok, first character of next word has been found */
  W[len++] = c;
  while (len<limit && (c=getchar())!=EOF && isalpha(c)) {
    /* 12+34 aWord is the first word */
    W[len++] = c;
  }
  /* now close off the string */
  W[len] = '\0';
  return 0;
}
```

Ex 7.16 and others

Combine Alistair's `getword.c` and `words.c` into one `.c` file, then change it to meet the requirement of Ex 7.16.

Implement

- 7.12 (medium),
- 7.14 (easy),
- 7.15 (a bit hard).

Another choice: group work - doing 7.12, 7.14 or some other exercises on board/paper

Unused materials

Strings

```
char s1[10]= “Hello”;
char *s2=“1234”;
```

Which of the following statements are OK:

```
1. s1++;
2. s2++;
3. s1= s2;
4. s2= s1 + 3;
```

Strings

```
typedef char word_t [11];
char *s="1234 abc9", *p= s;
word_t w, *q= w;
```

What the following fragments do?

1. `while (*p) *q++= *p++;`
2. `while (*q++= *p++);`
3. `while (isalpha(*p)) *q++= *p++;`
4.
```
while (!isalpha(*p)) p++;
  while (isalpha(*p)) *q++= *p++;
  *q= '\0';
```

Big-O (informal)

Equivalent writings:

1. $f(n) \in O(g(n))$
2. $f(n)$ grows no faster than $g(n)$
3. there is a constant $c > 0$ such that $c.g(n)$ is an upper bound of $f(n)$ when n is big enough

Strictly speaking:

$2n + 3 \in O(n) \in O(n \log n) \in O(n^2) \in \ldots$

But ***in computer science***, when using big-O we often mean ***the least upper bound***. So we only say:

$2n + 3 \in O(n)$

(Informal) Big-O Rules

Multiplicative constants can be reduced to 1:

$1000n^2$ or $0.0000001n^2$ is just n^2.

Base of logarithm doesn't matter:

$\log_{10} n$ or $\log_2 n$ is just $\log n$

Lower-level additive parts can be omitted:

$2n^3 + 100000n^2 + 6n + 10^{12}$ is just n^3.

Remember:

"better" algorithms

faster growing

1

$log(n)$ $(log\ n)^2$ ...

$\sqrt{n}$ n $n log(n)$ $n\sqrt{n}$ n^2 ...

... 2^n 2.5^n 3^n ...

$n!$

n^n

faster growing

Examples

Find big-O for:

a) $f_1(n) = 0.001n^3 + 10^9n^2 + 1$

b) $f_2(n) = 25\ (\log n)^5 + n + 100$

c) $f_3(n) = n^{0.1} + (\log n)^{10}$

d) $f_4(n) = (\log n)^3 + \sqrt{n}$

Program arguments

Write a program sum that accept two numbers and print out their sum. Example of execution:

```
$ ./sum 12 5
12.00 + 5.00 = 17.00
```

```
?: int main(int argc, char *argv[])
```

Program arguments: notes

Must check:

is the number of arguments as expected, and

when possible, is each argument valid.

Example: a program that accepts two positive numbers and print out their sum.

```
int main(int argc, char *argv[]) {
   double a, b;
   if (    argc != 3
        || (a=atof(argv[1]))<=0
        || (b=atof(argv[2]))<=0  ) {
      fprintf(stderr, "Usage: %s a b  where"
                 " a>0 and b>0\n",argv[0]);
       exit(EXIT_FAILURE);
   }
   printf("%.2f + %.2f = %.2f\n", a, b, a+b);
   return 0;
}
```