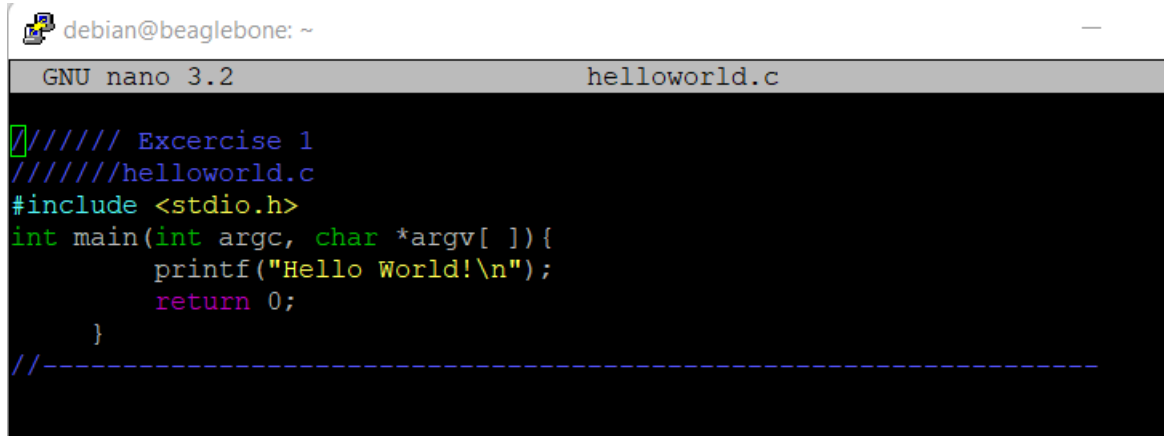


## Assignment II

### Exercise 1 : helloworld.c



```
debian@beaglebone: ~  
GNU nano 3.2 helloworld.c  
//////// Exercice 1  
////////helloworld.c  
#include <stdio.h>  
int main(int argc, char *argv[ ]){  
    printf("Hello World!\n");  
    return 0;  
}  
//-----
```

### Description

#include <stdio.h> // เรียกใช้งาน Library “Standard input/output”

int main(int argc, char \*argv[ ]){

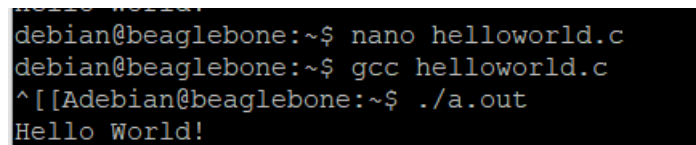
// เปิดใช้งานฟังก์ชันหลัก โดยที่กำหนดให้มี 2 argument โดยมี Argument Cout คือ argc ทำหน้าที่นับจำนวน Argument ที่รับเข้ามา และ Vector Argument หรือ argv ที่จะทำหน้าที่เป็น Pointer

printf("Hello World!\n"); // แสดงคำว่า “Hello World!” ออกทางหน้าจอ

return 0; // จบการทำงาน

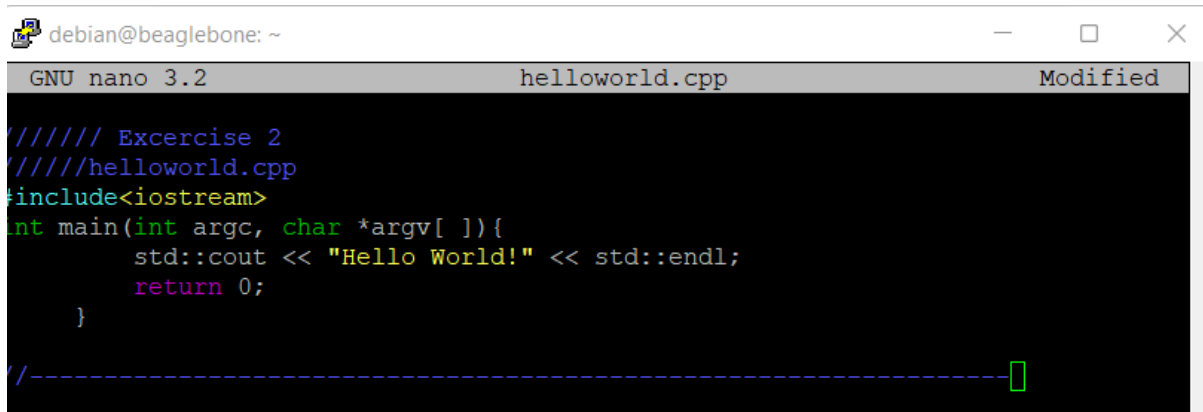
}

### การจำลองการทำงาน



```
debian@beaglebone:~$ nano helloworld.c  
debian@beaglebone:~$ gcc helloworld.c  
^[[Adebian@beaglebone:~$ ./a.out  
Hello World!
```

## Exercise 2 : helloworld.cpp



```

debian@beaglebone: ~
GNU nano 3.2 helloworld.cpp Modified
//////// Excercise 2
////////helloworld.cpp
#include<iostream>
int main(int argc, char *argv[ ]){
    std::cout << "Hello World!" << std::endl;
    return 0;
}
//-----

```

### Description

#include<iostream> // เรียกใช้งาน Library “iostream”

int main(int argc, char \*argv[ ]){

// เปิดใช้งานฟังก์ชันหลัก โดยที่กำหนดให้มี 2 argument โดยมี Argument Cout คือ argc ทำหน้าที่นับจำนวน Argument ที่รับเข้ามา และ Vector Argument หรือ argv ที่จะทำหน้าที่เป็น Pointer

std::cout << "Hello World!" << std::endl;

// แสดงข้อความ Hello World! ออกสู่หน้าจอด้วยคำสั่ง std::cout และใช้คำสั่ง std::endl เพื่อจบบรรทัดและขึ้นบรรทัดใหม่ หากใช้คำสั่งพร้อมกับ cout ต้องมี << คั่นเสมอ

return 0; // จบการทำงาน

}

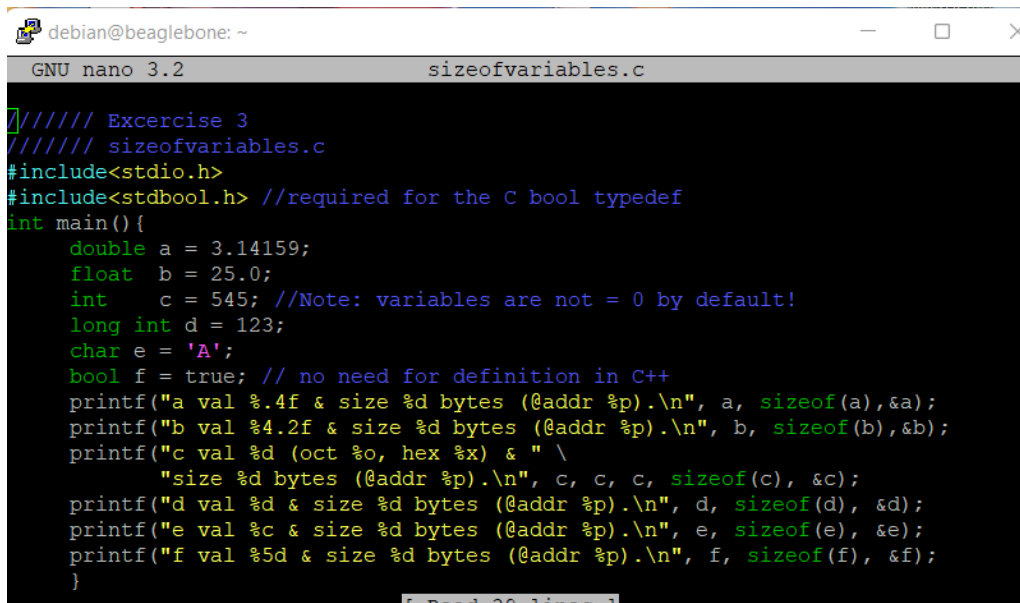
### การจำลองการทำงาน

```

debian@beaglebone:~$ nano helloworld.cpp
debian@beaglebone:~$ g++ helloworld.cpp
debian@beaglebone:~$ ./a.out
Hello World!
debian@beaglebone:~$

```

### Exercise 3 : sizeofvariables.c



```

debian@beaglebone: ~
GNU nano 3.2      sizeofvariables.c

// Excercise 3
// sizeofvariables.c
#include<stdio.h>
#include<stdbool.h> //required for the C bool typedef
int main() {
    double a = 3.14159;
    float b = 25.0;
    int c = 545; //Note: variables are not = 0 by default!
    long int d = 123;
    char e = 'A';
    bool f = true; // no need for definition in C++
    printf("a val %.4f & size %d bytes (@addr %p).\n", a, sizeof(a), &a);
    printf("b val %.2f & size %d bytes (@addr %p).\n", b, sizeof(b), &b);
    printf("c val %d (oct %o, hex %x) & " \
           "size %d bytes (@addr %p).\n", c, c, c, sizeof(c), &c);
    printf("d val %d & size %d bytes (@addr %p).\n", d, sizeof(d), &d);
    printf("e val %c & size %d bytes (@addr %p).\n", e, sizeof(e), &e);
    printf("f val %s & size %d bytes (@addr %p).\n", f, sizeof(f), &f);
}

```

#### Description

#include<stdio.h> // เรียกใช้งาน Library “stdio.h”

#include<stdbool.h> //required for the C bool typedef // เรียกใช้งาน Library “stdbool.h”

int main(){

double a = 3.14159; // ประกาศตัวแปร a เป็นชนิด double (ตัวแปรที่เก็บค่าทศนิยมได้ แต่มีพื้นที่สำหรับเก็บข้อมูลเยอะกว่า float 2 เท่า) ที่รับเก็บ 3.14159

float b = 25.0; // ประกาศตัวแปร b เป็นชนิด float (ตัวแปรที่เก็บค่าทศนิยมได้) ที่เก็บค่า 25.0

int c = 545; //Note: variables are not = 0 by default! //ประกาศตัวแปร c เป็นชนิด int ที่เก็บค่า 545

long int d = 123; //ประกาศตัวแปร d เป็นชนิด long int (เก็บข้อมูลได้มากกว่า int 2 bytes) ที่เก็บค่า 123

char e = 'A'; //ประกาศตัวแปร e เป็นชนิด char ที่เก็บค่า ‘A’

bool f = true; // no need for definition in C++

printf("a val %.4f & size %d bytes (@addr %p).\n", a, sizeof(a), &a);

//แสดงค่าของ a ด้วยทศนิยม 4 ตำแหน่ง (%.4f) และ ขนาดของข้อมูลแสดงในรูปของจำนวนเต็ม (%d) และ Address ของตัวแปร a แสดงตามตำแหน่งที่ Pointer ชี้ไปยังที่ตัวแปร a (%p) แล้วขึ้นบรรทัดใหม่ โดยที่นำค่ามาจาก a , sizeof(a) , address a

printf("b val %.2f & size %d bytes (@addr %p).\n", b, sizeof(b), &b);

//แสดงค่าของ b ด้วยหลักจำนวน 4 หลักและทศนิยม 2 ตำแหน่ง (%.2f) และ ขนาดของข้อมูลแสดงในรูปของจำนวนเต็ม (%d) และ Address ของ ตัวแปร d แสดงตามตำแหน่งที่ Pointer ชี้ไปยังที่ตัวแปร d (%p) แล้วขึ้นบรรทัดใหม่ โดยที่นำค่ามาจาก b , sizeof(b) , address b (& : Address Operator)

```
printf("c val %d (oct %o, hex %x) & " \
```

```
"size %d bytes (@addr %p).\n", c, c, c, sizeof(c), &c);
```

//แสดงค่าของ c เป็นจำนวนเต็ม (%d) และ แสดงค่าของ c ในรูปของฐานแปด (%o) และฐานสิบหก (%x) เนื่องจากไม่ต้องการให้คำสั่งยาวเกินไปจึงใช้คำสั่ง “ \ ” เพื่อบอกโปรแกรมว่าคำสั่งยังไม่จบ , แสดงขนาดของตัวแปร c ในรูปของ integer (%d) และ Address ของ ตัวแปร c แสดงตามตำแหน่งที่ Pointer ชี้ไปยังที่ตัวแปร c (%p) โดยที่นำค่ามาจาก c , sizeof(c) , address c (& : Address Operator)

```
printf("d val %d & size %d bytes (@addr %p).\n", d, sizeof(d), &d);
```

//แสดงค่าของ d เป็นจำนวนเต็ม %d และ ขนาดของข้อมูลแสดงในรูปของจำนวนเต็ม (%d) และ Address ของ ตัวแปร d แสดงตามตำแหน่งที่ Pointer ชี้ไปยังที่ตัวแปร d (%p) แล้วขึ้นบรรทัดใหม่ โดยที่นำค่ามาจาก d , sizeof(d) , address d (& : Address Operator)

```
printf("e val %c & size %d bytes (@addr %p).\n", e, sizeof(e), &e);
```

//แสดงค่าของ e เป็นตัวอักษร (char : %c) และ ขนาดของข้อมูลแสดงในรูปของจำนวนเต็ม (%d) และ Address ของ ตัวแปร e แสดงตามตำแหน่งที่ Pointer ชี้ไปยังที่ตัวแปร e (%p) แล้วขึ้นบรรทัดใหม่ โดยที่นำค่ามาจาก e , sizeof(e) , address e (& : Address Operator)

```
printf("f val %5d & size %d bytes (@addr %p).\n", f, sizeof(f), &f);
```

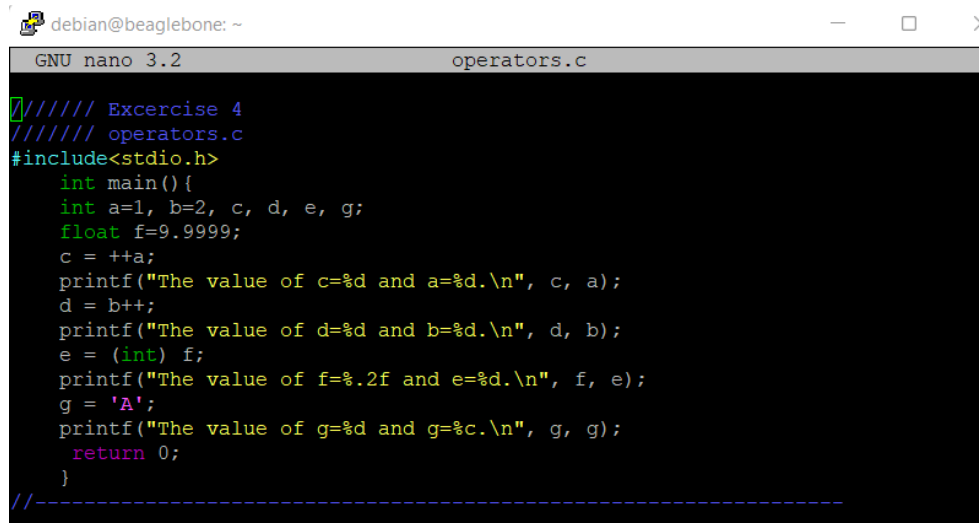
//แสดงค่าของ f เป็นจำนวนเต็ม 5 ตัว (%5d) หากไม่มีโปรแกรมจะทำการใส่ช่องว่างที่ข้างหน้าแทน และ ขนาดของข้อมูลแสดงในรูปของจำนวนเต็ม (%d) และ Address ของตัวแปร f แสดงตามตำแหน่งที่ Pointer ชี้ไปยังที่ตัวแปร f (%p) แล้วขึ้นบรรทัดใหม่ โดยที่นำค่ามาจาก f , sizeof(f) , address f (& : Address Operator)

```
}
```

### การจำลองการทำงาน

```
debian@beaglebone:~$ nano sizeofvariables.c
debian@beaglebone:~$ gcc sizeofvariables.c
debian@beaglebone:~$ ./a.out
a val 3.1416 & size 8 bytes (@addr 0xbe879560).
b val 25.00 & size 4 bytes (@addr 0xbe87955c).
c val 545 (oct 1041, hex 221) & size 4 bytes (@addr 0xbe879558).
d val 123 & size 4 bytes (@addr 0xbe879554).
e val A & size 1 bytes (@addr 0xbe879553).
f val 1 & size 1 bytes (@addr 0xbe879552).
debian@beaglebone:~$
```

## Exercise 4 : Operators.c



```

debian@beaglebone: ~
GNU nano 3.2 operators.c
//////// Excercise 4
//////// operators.c
#include<stdio.h>
int main(){
    int a=1, b=2, c, d, e, g;
    float f=9.9999;
    c = ++a;
    printf("The value of c=%d and a=%d.\n", c, a);
    d = b++;
    printf("The value of d=%d and b=%d.\n", d, b);
    e = (int) f;
    printf("The value of f=%.2f and e=%d.\n", f, e);
    g = 'A';
    printf("The value of g=%d and g=%c.\n", g, g);
    return 0;
}
//-----

```

### Description

#include<stdio.h> // เรียกใช้งาน Library “stdio.h”

int main(){ // เปิดใช้งาน main function

int a=1, b=2, c, d, e, g; //ประกาศตัวแปร a , b เป็นตัวแปรชนิด integer ที่เก็บค่า 1 และ 2 ตามลำดับ และ ตัวแปร c , d , e และ g เป็นตัวแปรชนิด integer

float f=9.9999; // ประกาศตัวแปร f เป็นตัวแปรชนิด float ที่เก็บค่า 9.9999

c = ++a; // ให้ตัวแปร c เก็บค่าจาก a+1

printf("The value of c=%d and a=%d.\n", c, a); // แสดง ค่าของ c และ a ในรูปของจำนวนเต็ม (integer : int : %d) แล้วขึ้นบรรทัดใหม่

d = b++; // ให้ตัวแปร d เก็บค่าจาก b แล้วบวกค่า b ไป 1

printf("The value of d=%d and b=%d.\n", d, b); // แสดง ค่าของ d และ b ในรูปของจำนวนเต็ม (integer : int : %d) แล้วขึ้นบรรทัดใหม่

e = (int) f; // ให้ตัวแปร e เก็บค่าของ f ในรูปของจำนวนเต็ม (integer : int)

printf("The value of f=%.2f and e=%d.\n", f, e); // แสดง ค่าของ f ในรูปของทศนิยม 2 ตำแหน่ง (%.2f) และ e ในรูปของจำนวนเต็ม (%d) แล้วขึ้นบรรทัดใหม่

g = 'A'; // ให้ตัวแปร g รับค่าของ A ที่เป็นรหัส ASCII (นำค่า A ไปเทียบใน ASCII table แล้วแสดงในรูปฐานสิบ)

printf("The value of g=%d and g=%c.\n", g, g); // แสดง ค่าของ g ในรูปของจำนวนเต็ม (%d) และ ค่าของ g ในรูปของตัวอักษร (Character : Char : %c) แล้วขึ้นบรรทัดใหม่

return 0; // จบการทำงาน

}

### การจำลองการแสดงผล

```
debian@beaglebone:~$ nano operators.c
debian@beaglebone:~$ gcc operators.c
debian@beaglebone:~$ ./a.out
The value of c=2 and a=2.
The value of d=2 and b=3.
The value of f=10.00 and e=9.
The value of g=65 and g=A.
```

## Exercise 5 : pointers.c

```

debian@beaglebone: ~
GNU nano 3.2 pointers.c
//////// Excercise 5
//////// pointers.c
#include<stdio.h>
int main(){
    int y = 1000;
    int *p;
    p = &y;
    printf("The variable has value %d and the address %p.\n", y, &y);
    printf("The pointer stores %p and points at value %d.\n", p, *p);
    printf("The pointer has address %p and size %d.\n", &p, sizeof(p));
    return 0;
}
//-----

```

### Description

#include<stdio.h> //เรียกใช้ฟังก์ชัน "stdio.h"

int main(){ // เปิดใช้งาน main function

int y = 1000; // ประกาศตัวแปร y ชนิด integer ที่รับค่า 1000

int \*p; // ประกาศตัวแปร p เป็น Pointer

p = &y; // กำหนดให้ p ที่ เป็น pointer ชี้ไปยัง Address ของตัวแปร y

printf("The variable has value %d and the address %p.\n", y, &y);

// แสดง ค่าของ y ในรูปของจำนวนเต็ม (%d) และ Address ของตัวแปร y ด้วย pointer (%p) แล้วขึ้นบรรทัดใหม่

printf("The pointer stores %p and points at value %d.\n", p, \*p);

// แสดง address ที่ p (Pointer) ชี้อยู่ และ ค่าที่เก็บอยู่ใน address นั้นๆ แล้วขึ้นบรรทัดใหม่

printf("The pointer has address %p and size %d.\n", &p, sizeof(p));

// แสดง address ที่ p (Pointer) ชี้อยู่ และขนาดของข้อมูล ณ ตำแหน่งนั้นๆ แล้วขึ้นบรรทัดใหม่

return 0; //จบการทำงาน

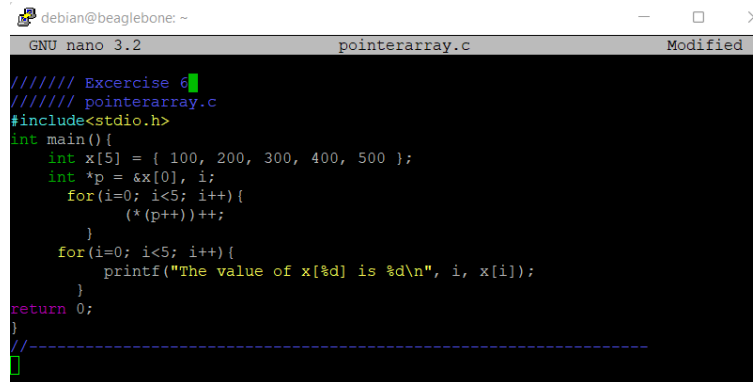
}

### การจำลองการทำงาน

```

debian@beaglebone: ~
GNU nano 3.2 pointers.c
//////// Excercise 5
//////// pointers.c
#include<stdio.h>
int main(){
    int y = 1000;
    int *p;
    p = &y;
    printf("The variable has value %d and the address %p.\n", y, &y);
    printf("The pointer stores %p and points at value %d.\n", p, *p);
    printf("The pointer has address %p and size %d.\n", &p, sizeof(p));
    return 0;
}
//-----

```

**Exercise 6 : pointerarray.c**


```

debian@beaglebone: ~
GNU nano 3.2 pointerarray.c Modified
//////// Excercise 6
//////// pointerarray.c
#include<stdio.h>
int main(){
    int x[5] = { 100, 200, 300, 400, 500 };
    int *p = &x[0], i;
    for(i=0; i<5; i++){
        (*(p++))++;
    }
    for(i=0; i<5; i++){
        printf("The value of x[%d] is %d\n", i, x[i]);
    }
    return 0;
}

```

Description

#include<stdio.h> // เรียกใช้งาน Library “stdio.h”

int main(){ // เปิดใช้งาน main function

int x[5] = { 100, 200, 300, 400, 500 }; // ประกาศ Array x ชนิด Integer ที่เก็บค่า 100 , 200 , 300 , 400 และ 500 ตามลำดับ

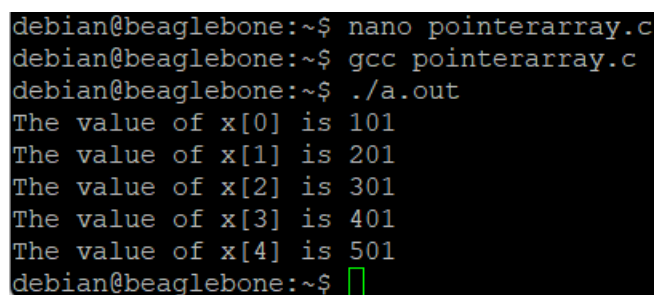
int \*p = &x[0], i; // กำหนดให้ p เป็น Pointer ชี้ไปยังตำแหน่ง Address 0 และประกาศตัวแปร i เป็นชนิด Integer  
 for(i=0; i<5; i++){ // กำหนดให้โปรแกรมวนลูป 5 รอบ  
 (\*(p++))++; // เพิ่มค่า ณ ตำแหน่งที่ pointer p ชี้อยู่ (x[0]) ไป +1 และทำการเลื่อนตำแหน่งของ pointer p ไป +1

}  
 for(i=0; i<5; i++){ // กำหนดให้โปรแกรมวนลูป 5 รอบ  
 printf("The value of x[%d] is %d\n", i, x[i]); //แสดง ค่าของ Array x ในรูปของจำนวนเต็ม (%d) แล้ว  
 ขึ้นบรรทัดใหม่  
 }

return 0; //จบการทำงาน

}

การจำลองการทำงาน



```

debian@beaglebone:~$ nano pointerarray.c
debian@beaglebone:~$ gcc pointerarray.c
debian@beaglebone:~$ ./a.out
The value of x[0] is 101
The value of x[1] is 201
The value of x[2] is 301
The value of x[3] is 401
The value of x[4] is 501
debian@beaglebone:~$ 

```



## Exercise 7 : cstrings.c

```

GNU nano 3.2
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main(){
    char a[20] = "hello ";
    char b[] = {'w','o','r','l','d','!','\0'}; // the \0 is important
    a[0]='H'; // set the first character to be H
    char *c = strcat(a,b); // join/concatenate a and b
    printf("The string c is: %s\n", c);
    printf("The length of c is: %d\n", strlen(c)); // call string length
    // find and replace the w with a W
    char *p = strchr(c,'w');
    // returns pointer to first 'w' char
    *p = 'W';
    printf("The string c is now: %s\n", c);
    if (strcmp("cat", "dog")<=0){ // ==0 would be equal
        printf("cat comes before dog (lexiographically)\n");
    }
    //insert "to the" into middle of "Hello World!" string - very messy!
    char *d = " to the";
    char *cd = malloc(strlen(c) + strlen(d));
    memcpy(cd, c, 5);
    memcpy(cd+5, d, strlen(d));
    memcpy(cd+5+strlen(d), c+5, 6);
    printf("The cd string is: %s\n", cd);
    //tokenize cd string using spaces
    p = strtok(cd, " ");
    while(p!=NULL){
        printf("Token:%s\n", p);
        p = strtok(NULL, " ");
    }
    return 0;
}

```

### Description

#include<stdio.h> //เรียกใช้งานฟังก์ชัน “stdio.h”

#include<string.h> //เรียกใช้งานฟังก์ชัน “string.h”

#include<stdlib.h> //เรียกใช้งานฟังก์ชัน “stdlib.h”

int main(){ // เปิดใช้งาน main function

char a[20] = "hello "; // กำหนด Array a รับค่า “hello”แบบ string

char b[] = {'w','o','r','l','d','!','\0'}; // the \0 is important //กำหนดตัวแปร array b รับค่า w , o , r , l และ \0 โดยที่ \0 หมายถึง null เป็นการระบุจบ Array

a[0]='H'; // set the first character to be H //กำหนดให้ Array a ตำแหน่งที่ 0 รับค่า H

char \*c = strcat(a,b); // join/concatenate a and b // กำหนดตัวแปร c เป็นตัวแปร Pointer มาชี้ ณ ตำแหน่งแรกที่เกิดจากการนำ Array b[] มาต่อท้าย Array a[]

printf("The string c is: %s\n", c); // แสดงข้อความ (%s) ที่ตัวแปร Pointer c ชี้อยู่ทั้งหมดจนกว่าจะเจอ \0 แล้วทำการขึ้นบรรทัดใหม่

printf("The length of c is: %d\n", strlen(c)); // call string length // แสดงความยาวของ c โดยใช้ฟังก์ชัน strlen ()

```
// find and replace the w with a W
char *p = strchr(c,'w'); // ให้ Pointer p กลับไปยังตำแหน่ง 'w' ตัวแรกที่โปรแกรมพบ ที่เกิดจากการนำ
Array b มาต่อท้าย array a โดยใช้ ฟังก์ชัน strchr ()
// returns pointer to first 'w' char
*p = 'W'; // ให้ Pointer p ชี้ไปยังตำแหน่งของ 'W'
printf("The string c is now: %s\n", c); // แสดงข้อความ (%s) ที่ตัวแปร Pointer c ชี้อยู่ทั้งหมดจนกว่าจะเจอ \0
แล้วทำการขึ้นบรรทัดใหม่

if (strcmp("cat", "dog")<=0){ // ==0 would be equal // ใช้คำสั่งเปรียบเทียบ (strcmp()) ถ้าตัวอักษรที่
เรานำไปเปรียบเทียบมาก่อนตัวเทียบจะให้ผลลัพธ์เป็น 1 แต่ถ้ามาทีหลังจะให้ผลลัพธ์เป็น -1 และถ้าเท่ากันหรือเหมือนกัน จะ
เป็น 0 ซึ่งในที่นี้ c มาก่อน d หากผ่านเงื่อนไข if ให้ไปบรรทัดต่อไป

printf("cat comes before dog (lexicographically)\n"); // แสดงข้อความ "cat comes before
dog (lexicographically)" แล้วขึ้นบรรทัดใหม่
}

//insert "to the" into middle of "Hello World!" string - very messy!
char *d = " to the"; // กำหนดให้ Pointer d เก็บข้อความ "to the" ซึ่งเป็นข้อความชนิด String
char *cd = malloc(strlen(c) + strlen(d)); //ทำการจองหน่วยความจำโดยใช้ฟังก์ชัน malloc()โดยมีขนาด
เท่ากับขนาดของ string ของ c และ d โดยหาความยาวได้จากฟังก์ชัน strlen() แล้วให้ Pointer cd ชี้ไปยังตำแหน่งดังกล่าว
memcpy(cd, c, 5); // ทำการคัดลอกข้อความจากตำแหน่งที่ c ชี้ มาใส่ยังตำแหน่งที่ cd ชี้ จำนวน 5 ตัว (เนื่องจาก
ตัวอักษร 1 ตัว กินพื้นที่ 5 bytes)
memcpy(cd+5, d, strlen(d)); // ทำการคัดลอกข้อความจากตำแหน่งที่ d ชี้ มาใส่ยังตำแหน่งที่ cd โดยทำการ
เลื่อนตัวชี้ไป +5 (cd+5) จำนวนเท่ากับขนาดของข้อความที่ Pointer d ชี้อยู่
memcpy(cd+5+strlen(d), c+5, 6); // ทำการคัดลอกข้อความจากตำแหน่งที่ c ชี้ มาใส่ยังตำแหน่งที่ cd ชี้อยู่
โดยเลื่อนตำแหน่งไป 5 + ความยาวของข้อความที่ Pointer d ชี้อยู่ จำนวน 6 ตัว
printf("The cd string is: %s\n", cd); // แสดงข้อความทั้งหมดที่ cd ชี้อยู่ จนกว่าจะเจอ \0 แล้วทำการขึ้น
บรรทัดใหม่

//tokenize cd string using spaces
p = strtok(cd, " "); //ทำการแบ่งข้อความโดยใช้คำสั่ง "strtok()"
while(p!=NULL){ // ทำการวนลูป เมื่อ p ยังไม่เจอ \0 หรือ วนลูปจนกว่าจะเจอ \0 ใน Array
printf("Token:%s\n", p); // แสดงข้อความ (%s) ที่ถูกแบ่ง แล้วขึ้นบรรทัดใหม่
p = strtok(NULL, " "); //ทำการแบ่งข้อความต่อจากครั้งก่อน โดยกำหนดให้ str เป็น NULL ไม่เช่นนั้น
โปรแกรมจะทำการแบ่งข้อความเดิมใหม่ ไม่ต่อจากที่ทำครั้งก่อนหน้า
}

return 0; // จบการทำงาน
}
```

### การจำลองการทำงาน

```
debian@beaglebone:~$ nano cstrings.c
debian@beaglebone:~$ gcc cstrings.c
debian@beaglebone:~$ ./a.out
The string c is: Hello world!
The length of c is: 12
The string c is now: Hello World!
cat comes before dog (lexiographically)
The cd string is: Hello to the World
Token:Hello
Token:to
Token:the
Token:World
debian@beaglebone:~$
```

Exercise 8 : makeLED.c

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define LED3_PATH "/sys/class/leds/beaglebone:green:usr3"
void writeLED(char filename[ ], char value[ ]); //function prototypes
void removeTrigger();
int main(int argc, char* argv[ ]){
    if(argc!=2){
        printf("Usage is makeLEDC and one of:\n");
        printf(" on, off, flash or status\n");
        printf(" e.g. makeLED flash\n");
        return 2;
    }
    printf("Starting the makeLED program\n");
    printf("The current LED Path is: " LED3_PATH "\n");
    // select whether command is on, off, flash or status
    if(strcmp(argv[1],"on")==0){
        printf("Turning the LED on\n");
        removeTrigger();
        writeLED("/brightness", "1");
    }
    else if (strcmp(argv[1],"off")==0){
        printf("Turning the LED off\n");
        removeTrigger();
        writeLED("/brightness", "0");
    }
    else if (strcmp(argv[1],"flash")==0){
        printf("Flashing the LED\n");
        writeLED("/trigger", "timer");
        writeLED("/delay_on", "50");
        writeLED("/delay_off", "50");
    }
    else if (strcmp(argv[1],"status")==0){
        FILE* fp; // see writeLED function below for description
        char fullFileName[100];
        char line[80];
        sprintf(fullFileName, LED3_PATH "/trigger");
        fp = fopen(fullFileName, "rt"); //reading text this time
        while (fgets(line, 80, fp) != NULL){
            printf("%s", line);
        }
        fclose(fp);
    }
    else{
        printf("Invalid command!\n");
    }
    printf("Finished the makeLED Program\n");
    return 0;
}

void writeLED(char filename[ ], char value[]){
    FILE* fp; // create a file pointer fp
    char fullFileName[100]; // to store the path and filename
    sprintf(fullFileName, LED3_PATH "%s", filename); // write path/name
    fp = fopen(fullFileName, "w+"); // open file for writing
    fprintf(fp, "%s", value); // send the value to the file
    fclose(fp); // close the file using the file pointer
}

void removeTrigger(){
    writeLED("/trigger", "none");
}

```

Description

#include<stdio.h> // เรียกใช้งาน Library “stdio.h”

#include<stdlib.h> // เรียกใช้งาน Library “stdlib.h”

#include<string.h> // เรียกใช้งาน Library “string.h”

#define LED3\_PATH "/sys/class/leds/beaglebone:green:usr3" //กำหนดเส้นทางของ LED3\_path

```

void writeLED(char filename[ ], char value[ ]); //function prototypes // สร้างฟังก์ชันที่ชื่อว่า writeLED
โดยภายในมี Parameters 2 ตัวที่เป็น Array คือ filename และ value
void removeTrigger(); // สร้างฟังก์ชันที่ชื่อว่า removeTrigger
int main(int argc, char* argv[ ]){ // เปิดใช้งานฟังก์ชันหลัก โดยที่กำหนดให้มี 2 argument โดยมี Argument Count
คือ argc ทำหน้าที่นับจำนวน Argument ที่รับเข้ามา และ Vector Argument หรือ argv ที่จะทำหน้าที่เป็น Pointer

    if(argc!=2){ // ถ้า argument ที่เข้ามานับได้ไม่ถึง 2 ให้ไปบรรทัดต่อไป

        printf("Usage is makeLEDC and one of:\n"); // แสดงข้อความ "Usage is makeLEDC and
one of " แล้วขึ้นบรรทัดใหม่

        printf(" on, off, flash or status\n"); // แสดงข้อความ "on, off, flash or status" แล้วขึ้นบรรทัด
ใหม่

        printf(" e.g. makeLED flash\n"); //แสดงข้อความ "e.g. makeLED flash" แล้วขึ้นบรรทัดใหม่
        return 2; //Return ค่า 2 ออกไปแล้วจบการทำงาน
    }

    printf("Starting the makeLED program\n"); // แสดงข้อความ "Starting the makeLED program"
แล้วขึ้นบรรทัดใหม่

    printf("The current LED Path is: " LED3_PATH "\n"); //แสดงจำนวนของเส้นทาง LED ในบรรทัดที่
ดึงมาจาก LED3_PATH

    // select whether command is on, off, flash or status
    if(strcmp(argv[1],"on")==0){ // ถ้าทำการเปรียบเทียบระหว่างค่าใน argv[1] และ string "on" เหมือนกัน
ให้ไปบรรทัดต่อไป

        printf("Turning the LED on\n"); //แสดงข้อความ "Turning the LED on" แล้วขึ้นบรรทัดใหม่
        removeTrigger(); //เรียกใช้ฟังก์ชัน removeTrigger()
        writeLED("/brightness", "1"); //เรียกใช้ฟังก์ชัน writeLED โดยกำหนดค่าที่จะส่งเข้าไปคือ
"/brightness" และ "1"
    }

    else if (strcmp(argv[1],"off")==0){ //ถ้าทำการเปรียบเทียบระหว่างค่าใน argv[1] และ String "Off"
เหมือนกัน ให้ไปบรรทัดต่อไป

        printf("Turning the LED off\n"); //แสดงข้อความ "Turning the LED off" แล้วขึ้นบรรทัดใหม่
        removeTrigger(); //เรียกใช้ฟังก์ชัน removeTrigger()
        writeLED("/brightness", "0"); //เรียกใช้ฟังก์ชัน writeLED โดยกำหนดค่าที่จะส่งเข้าไปคือ
"/brightness" และ "0"
    }

    else if (strcmp(argv[1],"flash")==0){ //ถ้าทำการเปรียบเทียบระหว่างค่าใน argv[1] และ String "flash"
เหมือนกัน ให้ไปบรรทัดต่อไป

```

```

printf("Flashing the LED\n"); //แสดงข้อความ "Flashing the LED" แล้วขึ้นบรรทัดใหม่
writeLED("/trigger", "timer"); //เรียกใช้ฟังก์ชัน writeLED โดยกำหนดค่าที่จะส่งเข้าไปคือ "/trigger"
และ "timer"

writeLED("/delay_on", "50"); //เรียกใช้ฟังก์ชัน writeLED โดยกำหนดค่าที่จะส่งเข้าไปคือ
"/delay_on" และ "50"

writeLED("/delay_off", "50"); //เรียกใช้ฟังก์ชัน writeLED โดยกำหนดค่าที่จะส่งเข้าไปคือ
"/delay_off" และ "50"
}

else if (strcmp(argv[1], "status") == 0) { //ถ้าทำการเปรียบเทียบระหว่างค่าใน argv[1] และ String
"status" เหมือนกัน ให้ไปบรรทัดต่อไป

FILE* fp; // see writeLED function below for description สร้างตัวแปรไฟล์ Pointer ชื่อ fp
char fullFileName[100]; //ประกาศตัวแปรชนิด char ชื่อ fullFileName ขนาด Array 100
char line[80]; //ประกาศตัวแปรชนิด char ชื่อ line ขนาด Array 80
sprintf(fullFileName, LED3_PATH "/trigger"); //นำเอาข้อความ "trigger" ไปใส่ต่อท้าย
LED3_PATH ที่เรา define ไว้ แล้วนำไปเก็บไว้ที่ fullFileName

fp = fopen(fullFileName, "rt"); //reading text this time // เปิดไฟล์ fullFileName เพื่ออ่าน
เพียงอย่างเดียว

while (fgets(line, 80, fp) != NULL) { //อ่านไฟล์จาก fp 80 ตัว มาเก็บที่ line โดยที่ถ้าไม่เท่ากับ
NULL ให้วนลูปต่อไป

printf("%s", line); //แสดงข้อความจาก line
}

fclose(fp); //ปิดไฟล์ที่ชื่ออยู่
}

else { //อื่นๆ

printf("Invalid command!\n"); // แสดงข้อความ "Invalid command!" แล้วขึ้นบรรทัดใหม่
}

printf("Finished the makeLED Program\n"); // แสดงข้อความ "Finished the makeLED Program"
return 0; //จบการทำงาน
}

void writeLED(char filename[ ], char value[]) { //ฟังก์ชัน writeLED

FILE* fp; // create a file pointer fp // สร้างตัวแปรไฟล์ Pointer ชื่อ fp
char fullFileName[100]; // to store the path and filename //สร้าง Array fullFileName ชนิด
char

```

```

sprintf(fullFileName, LED3_PATH "%s", filename); // write path/name //นำเอาข้อความ
“trigger” ไปใส่ต่อท้าย LED3_PATH ที่เรา define ไว้ แล้วนำไปเก็บไว้ที่ fullFileName
fp = fopen(fullFileName, "w+"); // open file for writing// เปิดไฟล์ fullFileName เพื่ออ่านเพียง
อย่างเดียว
fprintf(fp, "%s", value); // send the value to the file//นำค่าจาก value เขียนลงไปใน fp
fclose(fp); // close the file using the file pointer // ปิดไฟล์ที่ fp ซี่อยู่
}

void removeTrigger(){ //ฟังก์ชัน removeTrigger
writeLED("/trigger", "none"); //เรียกใช้ฟังก์ชัน writeLED แล้วส่งค่า “/trigger” และ “none” ไป
}

```

### การจำลองการทำงาน

```

debian@beaglebone:~$ nano makeLED.c
debian@beaglebone:~$ gcc makeLED.c
debian@beaglebone:~$ ./a.out
Usage is makeLEDC and one of:
  on, off, flash or status
  e.g. makeLED flash
debian@beaglebone:~$ echo $?
2
debian@beaglebone:~$ ./a.out on
Starting the makeLED program
The current LED Path is: /sys/class/leds/beaglebone:green:usr3
Turning the LED on
Finished the makeLED Program
debian@beaglebone:~$ echo $?
0
debian@beaglebone:~$ ./a.out off
Starting the makeLED program
The current LED Path is: /sys/class/leds/beaglebone:green:usr3
Turning the LED off
Finished the makeLED Program
debian@beaglebone:~$ echo $?
0
debian@beaglebone:~$ ./a.out flash
Starting the makeLED program
The current LED Path is: /sys/class/leds/beaglebone:green:usr3
Flashing the LED
Segmentation fault
debian@beaglebone:~$ echo $?
139
debian@beaglebone:~$ ./a.out status
Starting the makeLED program
The current LED Path is: /sys/class/leds/beaglebone:green:usr3
none rfkill-any rfkill-none kbd-scrolllock kbd-numlock kbd-capslock kbd-kanalock
kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock kbd-shiftllock kbd-shiftrl
ock kbd-ctrlillock kbd-ctrlrlock mmc0 mmc1 usb-gadget usb-host [timer] oneshot di
sk-activity disk-read disk-write ide-disk mtd nand-disk heartbeat backlight gpio
cpu cpu0 activity default-on panic netdev
Finished the makeLED Program
debian@beaglebone:~$ █

```

## Exercise 9 : bashLED

```

GNU nano 3.2
#!/bin/bash
LED3_PATH=/sys/class/leds/beaglebone:green:usr3
function removeTrigger
{
    echo "none" >> "$LED3_PATH/trigger"
}
echo "Starting the LED Bash Script"
if [ $# != 1 ]; then
    echo "There is an incorrect number of arguments. Usage is:"
    echo -e " bashLED Command \n where command is one of "
    echo -e " on, off, flash or status \n e.g. bashLED on "
    exit 2
fi
echo "The LED Command that was passed is: $1"
if [ "$1" == "on" ]; then
    echo "Turning the LED on"
    removeTrigger
    echo "1" >> "$LED3_PATH/brightness"
elif [ "$1" == "off" ]; then
    echo "Turning the LED off"
    removeTrigger
    echo "0" >> "$LED3_PATH/brightness"
elif [ "$1" == "flash" ]; then
    echo "Flashing the LED"
    removeTrigger
    echo "timer" >> "$LED3_PATH/trigger"
    echo "50" >> "$LED3_PATH/delay_on"
    echo "50" >> "$LED3_PATH/delay_off"
elif [ "$1" == "status" ]; then
    cat "$LED3_PATH/trigger";
fi
echo "End of the LED Bash Script"

```

### Description

```
#!/bin/bash
```

```
LED3_PATH=/sys/class/leds/beaglebone:green:usr3 //กำหนดเส้นทางของ LED_PATH
```

```
function removeTrigger //ฟังก์ชัน removeTrigger
```

```
{    echo "none" >> "$LED3_PATH/trigger"
```

```
//ใส่คำว่า "none" ลงไปที่ส่วนท้ายของ "$LED3_PATH/trigger"
```

```
}
```

```
echo "Starting the LED Bash Script" //แสดงข้อความ "Starting the LED Bash Script"
```

```
if [ $# != 1 ]; then //ถ้าจำนวน Argument ที่เพิ่มเข้ามาในสคริปต์นี้เกิน 1 ให้กระทำบรรทัดต่อไป
```

```
    echo "There is an incorrect number of arguments. Usage is:"
```

```
//แสดงข้อความ "There is an incorrect number of arguments. Usage is:"
```

```
    echo -e " bashLED Command \n where command is one of "
```

```
//แสดงข้อความ "bashLED Command \n where command is one of"
```

```
    echo -e " on, off, flash or status \n e.g. bashLED on "
```

```
//แสดงข้อความ "on, off, flash or status \n e.g. bashLED on"
```



```

    exit 2 // return ค่า 2 แล้วจบการทำงาน
fi //จบเงื่อนไข if
echo "The LED Command that was passed is: $1" // แสดงข้อความและ Argument ที่เข้ามา
if [ "$1" == "on" ]; then // ถ้าตรงตามเงื่อนไขให้ไปบรรทัดต่อไป
    echo "Turning the LED on" //แสดงข้อความ "Tming the LED on"
    removeTrigger // เรียกใช้งานฟังก์ชัน removeTrigger
    echo "1" >> "$LED3_PATH/brightness" //ใส่เลข 1 ลงไปต่อท้าย LED3_PATH/brightness
elif [ "$1" == "off" ]; then // ถ้าตรงตามเงื่อนไขให้ไปบรรทัดต่อไป
    echo "Turning the LED off" //แสดงข้อความ "Turning the LED off"
    removeTrigger // เรียกใช้ฟังก์ชัน removeTrigger
    echo "0" >> "$LED3_PATH/brightness" //ใส่เลข 0 ลงไปต่อท้าย LED3_PATH/brightness
elif [ "$1" == "flash" ]; then // ถ้าตรงตามเงื่อนไขให้ไปบรรทัดต่อไป
    echo "Flashing the LED" //แสดงข้อความ "Flashing the LED"
    removeTrigger // เรียกใช้ฟังก์ชัน removeTrigger
    echo "timer" >> "$LED3_PATH/trigger" //ใส่ timer ลงไปต่อท้าย LED3_PATH/trigger
    echo "50" >> "$LED3_PATH/delay_on" //ใส่ 50 ลงไปต่อท้าย LED3_PATH/delay_on
    echo "50" >> "$LED3_PATH/delay_off" //ใส่ 50 ลงไปต่อท้าย LED3_PATH/delay_off
elif [ "$1" == "status" ]; then // ถ้าตรงตามเงื่อนไขให้ไปบรรทัดต่อไป
    cat "$LED3_PATH/trigger"; //อ่านไฟล์จาก LED3_PATH/trigger
fi //จบเงื่อนไข if
echo "End of the LED Bash Script" //แสดงข้อความ "End of the LED Bash Script"

```

### การจำลองการทำงาน

เนื่องจากยังไม่สามารถจำลองผลได้ ต้องเพิ่มคำสั่ง `chmod u+x ____` ตามด้วยชื่อไฟล์ที่ต้องการ เพื่อให้สามารถประมวลผลไฟล์ได้ นั่นเอง และใช้ command `“./bashLED ____”` เพื่อจำลองการทำงาน

```
debian@beaglebone: ~
```

```
debian@beaglebone:~$ ./bashLED
Starting the LED Bash Script
There is an incorrect number of arguments. Usage is:
  bashLED Command
  where command is one of
  on, off, flash or status
  e.g. bashLED on
debian@beaglebone:~$ ./bashLED on
Starting the LED Bash Script
The LED Command that was passed is: on
Turning the LED on
End of the LED Bash Script
Starting the LED Bash Script
The LED Command that was passed is: on
Turning the LED on
End of the LED Bash Script
debian@beaglebone:~$ ./bashLED off
Starting the LED Bash Script
The LED Command that was passed is: off
Turning the LED off
End of the LED Bash Script
Starting the LED Bash Script
The LED Command that was passed is: off
Turning the LED off
End of the LED Bash Script
```

```
debian@beaglebone:~$ ./bashLED flash
Starting the LED Bash Script
The LED Command that was passed is: flash
Flashing the LED
./bashLED: line 26: /sys/class/leds/beaglebone:green:usr3/delay_on: Permission denied
./bashLED: line 27: /sys/class/leds/beaglebone:green:usr3/delay_off: Permission denied
End of the LED Bash Script
Starting the LED Bash Script
The LED Command that was passed is: flash
Flashing the LED
./bashLED: line 58: /sys/class/leds/beaglebone:green:usr3/delay_on: Permission denied
./bashLED: line 59: /sys/class/leds/beaglebone:green:usr3/delay_off: Permission denied
End of the LED Bash Script
debian@beaglebone:~$ ./bashLED status
Starting the LED Bash Script
The LED Command that was passed is: status
none rkill-any rkill-none kbd-scrolllock kbd-numlock kbd-capslock kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock
ck mmc0 mmc1 usb-gadget usb-host [timer] oneshot disk-activity disk-read disk-write ide-disk mtd nand-disk heartbeat backlight gpio cp
End of the LED Bash Script
Starting the LED Bash Script
The LED Command that was passed is: status
none rkill-any rkill-none kbd-scrolllock kbd-numlock kbd-capslock kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock
ck mmc0 mmc1 usb-gadget usb-host [timer] oneshot disk-activity disk-read disk-write ide-disk mtd nand-disk heartbeat backlight gpio cp
End of the LED Bash Script
```

การดัดแปลงคำสั่งเพื่อนำไปใช้ประโยชน์

```

debian@beaglebone: ~
GNU nano 3.2

#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <stdlib.h>
int main () {

printf("Hello Embedded World !"\
" i need Grade A\n");

int z;
float a[]={4,3.5,3,2.5,2,1.5,1,0};
float* p = &a[0];

for(z=0;z<8;z++)
{
printf("Grade you can get : %.2f\n",a[z]);
}
char b[]={ 'N','O','P','E','\t','\0'};
char c[10]="icant";
c[0]='I';
char*d=strcat(b,c);
printf("%s\n",d);
if(strcmp("A","B")>=0)
{
printf("You can get A (4.00)\n");
}
else if (strcmp("A","B")<0)
{
printf("Sorry dude,you should be retires btw\n");
}
unsigned char *e="but\t";
unsigned char *f=" i can help u.\0";
unsigned char *ef=malloc(strlen(e) + strlen(f));
memcpy(ef,e,strlen(e));
memcpy(ef+strlen(e),f,strlen(f));
printf("%s\n",ef);
unsigned char* h;
h=strtok(ef, " ");
while(h!=NULL)
{
printf("Token :%s\n",h);
h=strtok(NULL,"");
}
return 0;

```

ผมได้ทำการนำคำสั่งใน exercise 1-7 มาปรับใช้ โดยเป็นคำสั่งที่เกี่ยวกับการเกรด เหมือนการตอบโต้ของระบบ เพราะคำสั่งได้ถูก set คำตอบไปแล้ว แต่ก็ได้นำมาประยุกต์ใช้ หากทำการศึกษาต่อยอด ก็อาจจะมีการพิมพ์ตอบโต้กับระบบ เช่น ฉันอยากได้เกรด A แล้วระบบถามว่า อ่านหนังสือหรือยัง ถ้าเราตอบ yes ระบบจะ

ตอบกลับมาว่า คุณมีโอกาสได้ A แต่ถ้าตอบ No ระบบก็จะบอกคุณว่าเสีย F แต่ถ้าตอบว่า idk ระบบจะบอกว่า why you don't know . ประมาณนั้นครับ

#### การจำลองการทำงาน

```
debian@beaglebone:~$ nano helloembedded.c
debian@beaglebone:~$ gcc helloembedded.c
debian@beaglebone:~$ ./a.out
Hello Embedded World ! i need Grade A
Grade you can get : 4.00
Grade you can get : 3.50
Grade you can get : 3.00
Grade you can get : 2.50
Grade you can get : 2.00
Grade you can get : 1.50
Grade you can get : 1.00
Grade you can get : 0.00
NOPE      Icant
Sorry dude,you should be retires btw
but      i can help u.
Token :but
Token :i can help u.
debian@beaglebone:~$
```