/

| E | N | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|

SLIIT

Discover Your Future

Sri Lanka Institute of Information Technology

B.Sc. Eng. (Honours) / B.Eng. (Honours) Degree

End Semester Examination
Year 1, Semester II (2017)

# EC1441 – Engineering Programming

Duration: 2 Hour + 10 minutes reading time

November 2017

Instructions to Candidates:

- This is a **closed book** examination.
- This examination paper contains 16 pages including the cover page.
- This paper has **2 sections**. Answer **all questions.**
- Section 01: 20 MCQs, each question carries 3 marks for a total 0f 60 marks.
    - Correct answer for each question should be clearly marked in the given paper.
      *e.g.*
      *01 a [✓ b] c d*
- Section 02: you have to write **3 small programs using the given space**.
    - This section carries total of 40 marks.
- **Detach the answer sheet of Section 01 and the Section 02 from the examination paper.**
- **Only the answer sheet of Section 01 and the Section 02 should be submitted at the end of the examination.**
- Clearly **indicate your EN# on all the papers.**
- Total Mark is 100.
- Contributes 40% of the final grade.

| E | N | | | | | | | |
|---|---|---|---|---|---|---|---|---|

**SECTION 01:**
**Use the code given in each question and underline the most suitable answer.**
**(20 x 3 Marks)**

**Refer the code given below and answer the questions 01 and 02.**

```
#include <stdio.h>

void main(void)
{
        char ch[50];

        FILE *fp = fopen("test1.txt", "r");

        fgets(ch,10,fp);
        printf("%s \n", ch);
}
```

01. Select the **INCORRECT** statement.

a. **fopen** will return a **NULL** value in failing to open the file.
b. **test.txt** file is opened as a read only file.
c. **"r"** indicates mode of opening the file.
d. In the above code **fgets** will read the entire content of the file until the EOF.

02. If the "test1.txt" file is created in the same folder with the following content before running the program the above code will print:

**I am a first year engineering student.**

a. I am a first year engineering student.
b. I
c. I am a fi
d. The code will give a runtime error.

03.

```
#include <stdio.h>

int factorial (int number);
void main(void){
        int number;
        printf("Enter the number: ");
        scanf("%d", &number);
        factorial(number);
}
int factorial (int number) {
        int result = 0;
```

| E | N | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|

```
        if (number == 0)
                result = 1;
        else{
                result = number * factorial(number-1);
        }
        printf("Factorial: %d \n",result);

        return result;
}
```

Select the **INCORRECT** statement.

a. If the entered number is **5**, code will print **Factorial: 120.**
b. For any negative integer numbers entered, code will stop with a runtime error.
c. “factorial” function is an example for recursive functions.
d. Code will terminate with a runtime error.

04.

```
#include <stdio.h>
#include <string.h>

void main() {
        typedef struct Books {
          char title[50];
          char author[50];
          char subject[100];
          int book_id;
        }Book;

        Book book;

        strcpy( book.title, "C Programming");
        strcpy( book.author, "Nuha Ali");
        strcpy( book.subject, "C Programming Tutorial");
        book.book_id = 649;

        printf( "Book title : %s, ", book.title);
        printf( "Book author : %s, ", book.author);
        printf( "Book subject : %s, ", book.subject);
        printf( "Book ID : %d.\n", book.book_id);
}
```

Select the **INCORRECT** statement.

a. **typedef** is not a defined type in C programming.
b. **Books** is a type defined structure.

| E | N | | | | | | | |
|---|---|---|---|---|---|---|---|---|

c. **Book** is an alternative name for type **struct Books** which can be used to declare variables with the type struct Books.
d. Code will print,
Book title: C programming, Book author: Nuha Ali, Book subject: C Programming Tutorial,   Book ID: 649.

05.

```
#include <stdio.h>

void calc_square(int array1[]){
        int i;
        for (i=0; i<5; i++)
                array1[i] *= array1[i];
}
int main(void) {
        int array1[] = {1,2,3,4,5};
        int i;
        calc_square(array1);

        for (i=0; i<5; i++)
                printf("%d ",array1[i]);
        return 0;
}
```

The above code will print:

a. 1 4 9 16 25
b. 1 2 3 4 5
c. 0 1 2 3 4
d. Compile error

06.

```
#include <stdio.h>
int arraySum (int *array, const int n);
void main () {

        int values[10] = {3, 7, -9, 3, 6, -1, 7, 9, 1, -5};

        printf("The sum is %i \n", arraySum(values, 10));
}
int arraySum (int *array, const int n) {

        int sum = 0, *ptr;
        int *const arrayEnd = array + 9;

        ptr = array;
        while(ptr <= arrayEnd) {
                sum += *ptr;
                ptr++;
        }
        return sum;
}
```

The above code will print:

a. The sum is 0.
b. Set of address values.
c. The sum is 21.
d. Code will give a compile error.

**Using the code given below answer questions 07 and 08.**

```
#include <stdio.h>
#include <string.h>

void main(void) {
        char x[] = "Hello";
        char y[] = "World";
        char z[] = "Hello";
        char a[20];

        strcpy(a,"Test");
        strcat(a,x);

        printf("Compare result: %d \n", strcmp(x,z));
        printf("Length of string y: %d \n", strlen(y));
}
```

07. Select the **CORRECT** statement.

a. **'strcmp'** function will return 0, if both string values provided are exactly matching.
b. **'strlen'** function will give the length of a string value including '\0'.
c. **'strcpy'**, **'strcmp'**, **'strcat'** and **'strlen'** functions are defined in the stdio header file.
d. **'strcat'** function will concatenate the two string values and the resultant string will be stored variable x.

08.
The above code will print:

a. Compare result: 0
   Length of string y: 6
b. Compare result: 1
   Length of string y: 6
c. Compare result: 0
   Length of string y: 5
d. Will give a compile error.

| E | N | | | | | | | |
|---|---|---|---|---|---|---|---|---|

09.

```
#include <stdio.h>
void calc_square(int *ptr);
int main(void) {
        int array1[] = {1,2,3,4,5};
        int i;
        calc_square(array1);

        for (i=0; i<5; i++)
                printf("%d ",array1[i]);
        return 0;
}
void calc_square(int *ptr){
        int i;
        for (i=0; i<5; i++) {
                (*ptr)++;
                ptr++;
        }
}
```

The above code will print:

a. 1 2 3 4 5
b. 2 3 4 5 6
c. Prints set of address values.
d. Compile error

10.

```
#include <stdio.h>

float Average(int *a, int *b, int *c){

        float avg = (*a+*b+*c)/3.0;
}

void main (void) {
        int a, b, c;

        printf("Enter the numbers: ");
        scanf("%d %d %d", &a, &b, &c);

        printf("Average: %.2f \n", Average(&a,&b,&c));
}
```

Select the **CORRECT** statement.

a. Code will print a garbage value.
b. If the entered values for a, b and c are 3, 5 and 7 the code will print **Average: 7.50**

09.

```
#include <stdio.h>
void calc_square(int *ptr);
int main(void) {
        int array1[] = {1,2,3,4,5};
        int i;
        calc_square(array1);

        for (i=0; i<5; i++)
                printf("%d ",array1[i]);
        return 0;
}
void calc_square(int *ptr){
        int i;
        for (i=0; i<5; i++) {
                (*ptr)++;
                ptr++;
        }
}
```

The above code will print:

a. 1 2 3 4 5
b. 2 3 4 5 6
c. Prints set of address values.
d. Compile error

10.

```
#include <stdio.h>

float Average(int *a, int *b, int *c){

        float avg = (*a+*b+*c)/3.0;
}

void main (void) {
        int a, b, c;

        printf("Enter the numbers: ");
        scanf("%d %d %d", &a, &b, &c);

        printf("Average: %.2f\n", Average(&a,&b,&c));
}
```

Select the **CORRECT** statement.

a. Code will print a garbage value.
b. If the entered values for a, b and c are 3, 5 and 7 the code will print **Average: 7.50**

| E | N | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|

c. Code will give a compile error.
d. Code will terminate with a runtime error.

11.

```
#include <stdio.h>
int c = 10;                        //global variable declaration
int main () {

        int c = 30;                //local variable declaration

        printf("c = %d \n", c);
        return 0;
}
```

The above code will print:

a. c = 10
b. c = 30
c. Code will give a compile error.
d. Code will terminate with a runtime error.

12.

```
#include <stdio.h>
#include <string.h>

union Data {
  int i;
  float f;
  char str[20];
};
int main(void) {

  union Data data;
  data.i = 10;
  data.f = 220.5;
  strcpy( data.str, "C Programming");

  printf( "data.i : %d, ", data.i);
  printf( "data.f : %f, ", data.f);
  printf( "data.str : %s\n", data.str);
}
```

Select the **INCORRECT** statement.

a. **Union** is structure type which allocate a single memory space.
b. Data cannot be assigned for all the members at the same time.

| E | N | | | | | | | |
|---|---|---|---|---|---|---|---|---|

c. The code will print:
data.i : 10, data.f : 220.500000, data.str : C Programming

d. **Union** gives the advantage of using a single memory space to store values with different data types.

13.

```
#include <stdio.h>

int main (void) {

        int ch = "EP EC1441";
        char *ptr;
        ptr = a;
        printf("%s \n", ptr);
        return 0;
}
```

The above code will print:

a. EP
b. EP EC1441
c. EC1441
d. Code will terminate with a runtime error.

14.

```
#include <stdio.h>

void exchange (int *ip1, int *ip2);

void main (void) {

        int a = 10;
        int b = 20;
        int *point_a, *point_b;

        point_a = &a;
        point_b = &b;
        printf("a = %d, b = %d\n", a, b);

        exchange(&a,&b);
        printf("a = %d, b = %d\n", a, b);
}
void exchange (int *ip1, int *ip2){
        int temp = 0;
        temp= *ip1;
        *ip1 = *ip2;
        *ip2 = temp;
}
```

| E | N | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|

The above code will print:

a. a = 10, b = 20
b. a = 10, b = 20
   a = 20, b = 10
c. a = 20, b = 20
   a = 10, b = 10
d. a = 20, b = 10
   a = 10, b = 20

15.

```
#include <stdio.h>

struct user {
        int UserId;
        char name[20];
        int age;
        float weight;
};
void main(void) {

        struct user employee1 = {20, "eric", 22, 45.8};

        printf("Employee ID: %d, ", employee1.UserId);
        printf("Name: %s, ", employee1.name);
        printf("Age: %d, ", employee1.age);
        printf("Weight: %.2f \n", employee1.weight);
}
```

The above code will print:

a. Employee ID: 22, Name: "eric", Age: 20, Weight: 45.80.
b. Employee ID: 20, Name: eric, Age: 22, Weight: 45.80.
c. Employee ID: 20, Name: Eric, Age: 22, Weight: 45.
d. Code will give a compile error: 'struct' undeclared identifier.

16.

```
#include <stdio.h>

void main(void) {
        struct date {
                int day;
                int month;
                int year;
        };
        struct date today;
        struct date *datePtr;
        datePtr = &today;
```

| E | N | | | | | | | |
|---|---|---|---|---|---|---|---|---|

```
        datePtr -> day = 06;
        datePtr -> month = 11;
        datePtr -> year = 2017;
        printf("Today's Date is: %d %d %d \n",
                datePtr->day, datePtr->month, datePtr->.year);
}
```

Select the **INCORRECT** statement.

a. The code will print:
   Today's Date is: 06 11 2017
b. The code will give a compile error.
c. datePtr is structure type pointer variable.
d. **(*datePtr).day** notation gives the same meaning as **datePtr -> day**

17.

```
#include <stdio.h>

void main() {
        int i = 0;
        struct entry {
                int val;
                struct entry *next;
        }n1, n2, n3, n2_3

        struct entry *listPtr = &n1;

        n1.val = 100;   n2.val = 200;
        n3.val = 300;   n2_3.val = 250;

        n1.next = &n2;
        n2.next = &n3;
        n2_3.next = n2.next;
        n2.next = &n2_3;
        n3.next = (struct entry *) 0;

        while(listPtr != (struct entry *) 0) {
                printf("%d, ", listPtr -> val);
                listPtr = listPtr -> next;
        }
}
```

The above code will print:

a. 100, 200, 300, 250
b. 300, 200, 250, 100
c. 100, 200, 250, 300
d. The code will give a runtime error.

| E | N | | | | | | | |
|---|---|---|---|---|---|---|---|---|

18.

```
#include <stdio.h>

void main() {
        typedef int count;
        count dcount = 5;

        while(dcount) {
                printf("%d ", dcount--);
        }
}
```

The above code will print:

a. 5 4 3 2 1
b. 5 4 3 2 1 0
c. 1 2 3 4 5
d. The code will give a compile error.

19.

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
  int num = 2, i, *ptr, sum = 0;

  ptr = (int*) calloc(num, sizeof(int));

  printf("Enter elements of array: ");
  for(i = 0; i < num; i++)
  {
    scanf("%d", ptr + i);
    sum += *(ptr + i);
  }
  printf("Sum = %d", sum);
  free(ptr);
  return 0;
}
```

Select the **CORRECT** statement.

a. **calloc** is not a function used for dynamic memory allocation.
b. **calloc** requires two input arguments. Those are:
   number of memory spaces required and the size of a single memory space.
c. The dynamically allocated memory will be free automatically with the termination of the program.
d. If the dynamic memory allocation is not successful **calloc** will return the value as **1**.

| E | N | | | | | | | |
|---|---|---|---|---|---|---|---|---|

20. A student is developing a code to read names of 10 different students from a user. Observe the lines given in the code and select the suitable answer for the missing line to perform the required task.

```
#include <stdio.h>

void main() {

  ...............................
  int i = 0;

  for(i=0;i<10;i++) {
      printf("Enter the name %d: ", i+1);
      gets(Names[i]);
  }
}
```

The above code will print:

a. char Names[10];
b. char *Names[10];
c. char Names[10][20];
d. char Names[20][10];

*** End of SECTION 01 ***

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| E | N | | | | | | | | |

**ANSWER SHEET for SECTION 01:**

Correct answer of each question should be clearly marked as in the example given below.

*e.g.*
*01* *a* [✓ *b*] *c* *d*

---

| | | | | |
|---|---|---|---|---|
| 01. | a | b | c | d |
| 02. | a | b | c | d |
| 03. | a | b | c | d |
| 04. | a | b | c | d |
| 05. | a | b | c | d |
| 06. | a | b | c | d |
| 07. | a | b | c | d |
| 08. | a | b | c | d |
| 09. | a | b | c | d |
| 10. | a | b | c | d |
| 11. | a | b | c | d |
| 12. | a | b | c | d |
| 13. | a | b | c | d |
| 14. | a | b | c | d |
| 15. | a | b | c | d |
| 16. | a | b | c | d |
| 17. | a | b | c | d |
| 18. | a | b | c | d |
| 19. | a | b | c | d |
| 20. | a | b | c | d |

| E | N | | | | | | | |
|---|---|---|---|---|---|---|---|---|

**SECTION 02:**
**Write the answer to each question using the given space.**

01. Write a simple program to calculate and display the $n^{th}$ triangular number.

*Hint: n is a user given integer number. Where, $1 \le n \le 50$*

(10 Marks)