UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 1, Examination, Dec 2023

Module No: EE1101   **Module Name: Computer Programming I (c18)**

Part I

Instructions for candidates

- Write your index number on top of every page.

- Question paper contains 50 multiple choice questions.

- Each question carries 0.5 marks.

- Answer all questions. Each question has only one answer.

- **For each question, put an X mark on the letter: (a), (b), (c), or (d) which corresponds to the correct answer, by using a black or blue pen.**

- Time allowed is **1 hour and 30 minutes.**

1. Founder of the C language is

    (a) Richard Stallman
    (b) Dennis Ritchie
    (c) Bjarne Stroustrup
    (d) Linus Torvalds

2. Minimum program in C is

    (a) `main(void){}`
    (b) `int main(void){return 0;}`
    (c) `void main(){}`
    (d) `main(){}`

3. What is false in Codeblock

    (a) Workspace may contain more than one Project
    (b) Project may contain more than one source file.
    (c) Project is always included in a Workspace
    (d) Both Workspace and Project are file folders

4. In Codeblock project is named `Prj` Then the folder

    (a) `..\Prj\obj\` contains all source files
    (b) `..\Prj\bin\` contains all object files
    (c) `..\Prj\bin\Debug\` contains the executable file
    (d) `..\Prj\` contains the file `main.o`

5. Which is false on Codeblock?

    (a) *Build* is equivalent to *compile* and *link*
    (b) *Watch* allows to monitor only variables while debugging
    (c) *Run* executes the program
    (d) *Debug* stops execution at *break points*

6. The code snippet

    ```
    float x,y;
    scanf("%f",&x); scanf("%f", &y);
    printf("%f", x/y);
    ```

    (a) contains a syntax error
    (b) may generate run time error
    (c) can not be debug
    (d) produces unpredictable result

7. A program with a *compile time error*

    (a) can not be executed.

(b) can be linked with other library files.

(c) can be executed, but an error message will be displayed.

(d) will produce wrong results during execution.

8. Which of the following situations will not generate a *segmentation fault*

(a) Accessing data segment

(b) Accessing out-of-array index bounds

(c) Improper use of `scanf()`

(d) Stack overflow

9. Statements are terminated with

(a))

(b) ;

(c) "

(d) \n

10. A compound statement is made by using

(a) (){}

(b) []

(c) <>

(d) {}

11. Which of the followings is *not* a keyword?

(a) enum

(b) typedef

(c) automatic

(d) static

12. In `printf("Hello %d", 34);`

(a) "Hello %d" is a formatted C-String

(b) the literal constant 34 should be replaced with a variable to avoid errors

(c) %d is inappropriate format specifier.

(d) `printf()` is included in `stdio.h`

13. In `scanf("%d", &x);`

(a) %d is the address of d

(b) &x is format specifier of x

(c) `scanf()` is a user defined function

(d) x is accessible to `scanf()`

14. What is not true on C language syntax?

(a) Indentation indicates a block of code

(b) Newlines and tabs are considered white spaces and are ignored

(c) Statements ends in semicolon (;)

(d) Variable names may start with underscore (eg `int _x;`)

15. `int x='A';printf("%x", x);`

The code snippet displays 41. Therefore

(a) 41 must be the exit code

(b) 14 is how 'A' is type cast to x

(c) ASCII code of A is 65

(d) ASCII code of A is 41

16. What is the key difference between `signed int` and `unsigned int`?

(a) `signed int` can store negative values, while `unsigned int` cannot.

(b) `unsigned int` can store larger values than `signed int`.

(c) Both `signed int` and `unsigned int` have the same range of representable values.

(d) `signed int` is always 64 bits, and `unsigned int` is always 32 bits.

17. The code snippet displays

```
int x=789,y;
printf("%d", y=x-- - --x);
```

(a) 788

(b) 0

(c) 2

(d) syntax error

18. Type declaration `int x;` defines

(a) the size of the x as 4 bytes in 64 byte processor machines.

(b) the operations on x to be floating point category.

(c) possible pointers and references for data.

(d) the memory address of x as *x.

19. C variable types that best fits to store a student's name and OGPA are

(a) `char` and `float`.

(b) array of `char` and `float`.

(c) array of `pointer` and `void`.

(d) `String` and `float`.

20. Which of the statements makes `int *ptr;` points to `int x;`?

    (a) `*ptr = x;`

    (b) `*ptr !=x;`

    (c) `ptr[x]=*x;`

    (d) `ptr = &x;`

21. Which of the statements generate a syntax error, if
`int *ptr, x[]={1,2,3,4,5};`

    (a) `ptr = a;`

    (b) `x[0] = x[1];`

    (c) `x[2] = *(++x));`

    (d) `x[3] = *(x+1));`

22. If `int x[]={12,13,14,15},*p=&x[2];` then `printf("%d",*(p-1));` displays

    (a) 12

    (b) 13

    (c) 14

    (d) unpredictable value

23. Which of the format specifiers is used to print the values of a double?

    (a) `%lf`

    (b) `%d`

    (c) `%s`

    (d) `%c`

24. The operator with the highest precedence among the following is

    (a) `()`

    (b) `[]`

    (c) `?:`

    (d) `>>=`

25. Operators with lowest precedence are

    (a) conditional operators

    (b) logical operators

    (c) assignment operator

    (d) bit-wise operators

26. The statement `(5/9)*(f-32);` evaluates to

    (a) 8.3 if `f` is set to 50.

    (b) non zero value, based on the value of `f`

    (c) 0

    (d) a value depending on the type of `f`

27. Which is not a preprocessor directive?

    (a) `define`

    (b) `goto`

    (c) `include`

    (d) `ifndef`

28. What is given in the following?
`int fun[10][11];`

    (a) Declaration of two dimensional array `fun`

    (b) Definition of the operator `[ ][ ]`.

    (c) Type casting of `int`.

    (d) Definition of function `fun`

29. Which is the *carriage return* character?

    (a) `'\n'`

    (b) `'\r'`

    (c) `'\t'`

    (d) `'\0'`

30. What is the meaning of `!x != y;` ?

    (a) Is x not equals y?

    (b) Is not-x not-equals y?

    (c) Why y is not equals not-x?

    (d) Is x is not equal to y?

31.
```
for(i=0;i<5;++i){                 1
    for(j=0;j<5;++j)              2
        if(j<i)   printf(" ");    3
        else      printf("*");    4
    printf("\n");                 5
}                                 6
```

Does this code snippet print the following?
```
*****
****
***
**
*
```

(a) Yes

(b) No

(c) Prints only if j<=5 in line no 2

(d) Prints only if i<j in line no 3

32. `if((x<13 || x>27) && (x<-7 || x>-29)) printf("Yes");`

This will not display YES if x is

(a) 33

(b) 51

(c) -43

(d) 17 .

33. `(12==5 && 3!=3) || (4+5 || 3-4+1)`

This expression evaluates to

(a) -1

(b) 0

(c) 1

(d) unpredictable value

34.
```
if(x<0) y=1;
else if(x<10) y=2;
else y=3;
```

If x is 8 then y becomes

(a) 1

(b) 2

(c) 3

(d) 0

35.
```
if(age<65) printf("Senior");
else if(age<25) printf("Middle");
else if(age<5) printf("Young");
else printf("Kid");
```

This code snippet displays Kid, if age is

(a) 4

(b) 15

(c) 37

(d) 75

36. The code `for(;;);`

(a) generates compile time error

(b) has incorrect syntax

(c) has logical errors

(d) runs forever

37. `for(i=91;i>78;i-=3) printf("X");`
How many times the character X is displayed?

(a) 6

(b) 5

(c) 4

(d) 0

38.
```
int i, j;
for (i = 1; i <= 3; i++) {
    for (j = 3; j >= 1; j--) {
        if (i == j) break;
        printf("%d%d ", i, j);
    }
}
```

What is the output of this code snippet?

(a) 12 13 23

(b) 13 12 23

(c) 13 12 32

(d) 31 21 32

```
int i=8;
while(i-->0)
{  printf("%d",i%5);}
```

This code snippet outputs

(a) 21032103

(b) 21043210

(c) 21432143

(d) syntax error

```
int i=8,j=6,k=-1;
while(i!=j){
    printf("%d%d:",i,j);
    i -= 4; j += k*3;
}
```

This code snippet outputs

(a) Nothing

(b) Unpredictable content as the loop runs forever

(c) 86:43:

(d) 86:43:00:

39.
```
int x=1, total=0,y;
while(x<=3){
    y=x*x; printf("%d",y);
    total +=y; ++x;
}
printf("%d",total);
```

The above code snippet displays

(a) 14914

(b) 14941

(c) 14491

(d) 11942

```
i=2;
do{ printf("%d ",i); }
while(i--);
```

This code snippet outputs

(a) 2 1 0

(b) 210

(c) 2

(d) syntax error

40. What outputs the code snippet if j=4?
```
switch (x){
    case 7:j*=3;
    case 9:j*=4;break;
    case 5:j*=5;break;
}
printf("%d", j);
```

(a) 20, if x is set to 9

(b) 16, if x is set to 5

(c) 36, if x is set to 9

(d) 48, if x is set to 7

41.
```
int i;
for(i=10;i>0;i--)
if((i/2)*2==i) printf("%d ", i);
```

The above code snippet displays

(a) 10 8 6 4 2

(b) 10 8 6 4 2 0

(c) 9 7 5 3 1

(d) 0 0 0 0 0

42.
```
int isPrime(int num){
    int i;
    if (num < 2) return 0;
    for (i = 2; i <= sqrt(num);
      i++) {
      if (num % i == 0) return 0;
    }
    return 1;
}
```

The intended purpose of isPrime() is return 1 when nuum is a prime number. The code of isPrime() is

(a) correct

(b) incorrect

(c) correct when num is less than 100000

(d) incorrect because it tests if num is a perfect number.

43.
```
int foo(const char p[]) {
    int m = 0;
    while (p[m] != '\0') m++;
    return m;
}
```

This function foo() returns

(a) m increased by 1 if p[m] is '\0'

(b) returns the length of c-string p.

(c) the number of constants in p[]

(d) the number of null characters in p[]

44. `void pow(void);`
is a function

(a) definition

(b) prototype

(c) call

(d) name

45. According to
`double xy(int x, float y);`
the return value is

(a) of type float.

(b) of type int.

(c) of type double.

(d) of non of the types given above.

46. Local variables of a function

(a) can be define during run-time.

(b) sometimes can act as global variables.

(c) do exist only during the function execution.

(d) are constants and can not be changed.

47. What would be z, if z=foo(8);, where

```
int foo(int x){return x*x*x;}
```

(a) 8

(b) 512

(c) 888

(d) 24

48.
```
double d1, d2, *p1, *p2;
p1 = &d1; p2 = &d2;
*p1=7; *p2=8;*p1=*p1**p2;
p1 = &d2; p2 = &d1;
*p1=*p1**p2;
printf("%.0lf", *p1);
```

The code snippet outputs

(a) 531

(b) 448

(c) 243

(d) 1030

49.
```
int i=0;
float *p, x[]={1,2,3,4,5};
p=x;
while(*p!=5) { *p *= *p; ++p;}
printf("%f",x[4]);
```

The code snippet outputs

(a) 5

(b) 25

(c) 26

(d) 16

50.
```
int foo(int n) {
  if (n == 0 || n == 1) return 1;
  else return n * foo(n - 1);
}
```
The function \lstinline|foo()|

(a) returns a pointer to foo

(b) is a recursive function

(c) is a delegate function

(d) allocates memory dynamically