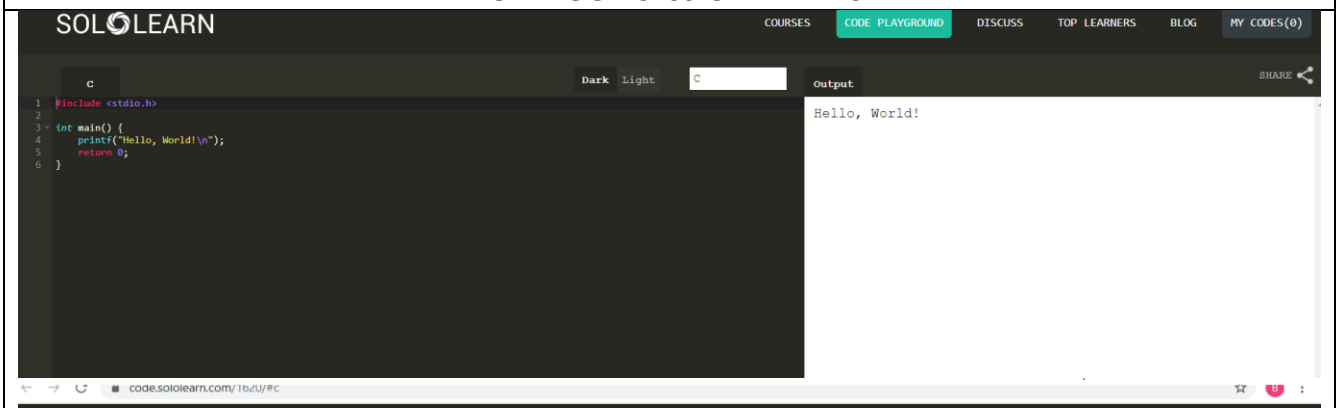
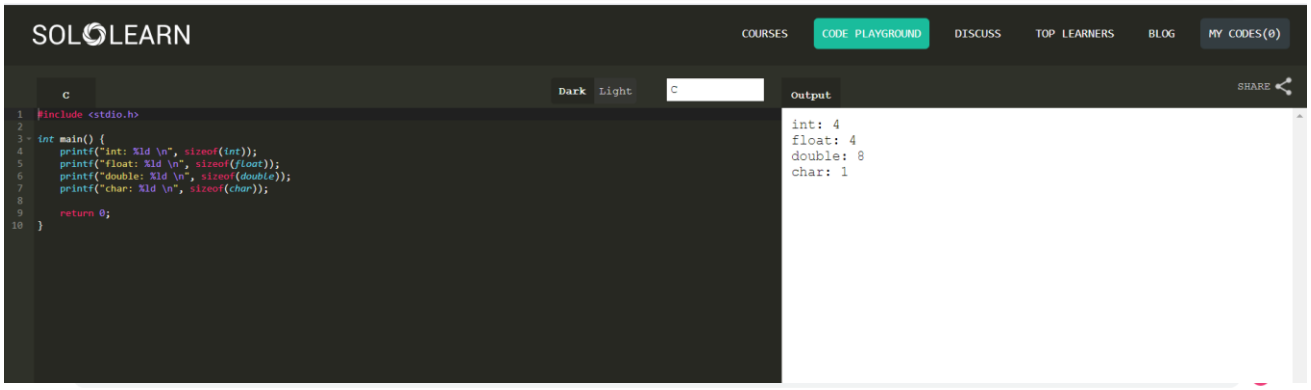
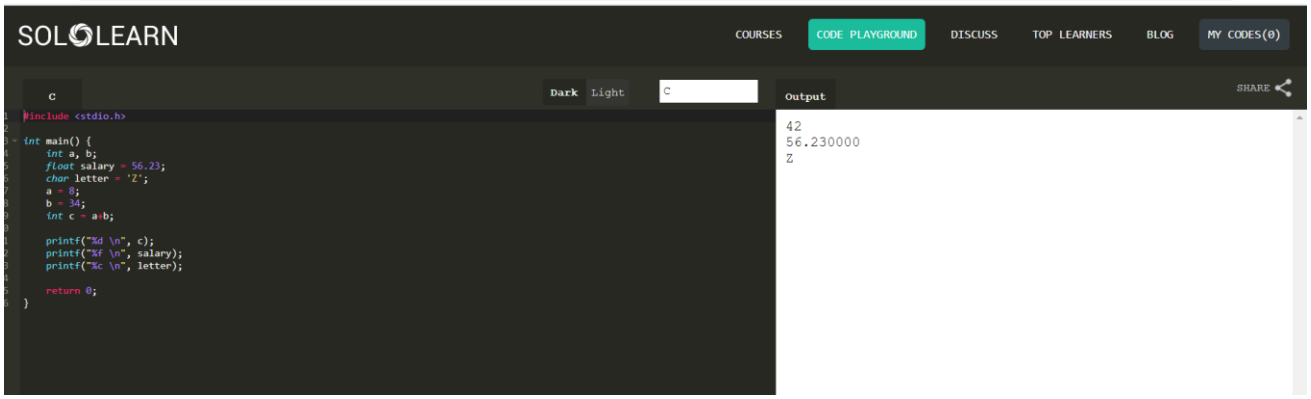
DAILY ASSESSMENT FORMAT

Date:	**18-06-2020**	**Name:**	**BINDUSHRI**
Course:	**C programming**	**USN:**	**4AL17EC011**
Topic:	**Basic concepts** **Data types**		**6th A**
Github Repository:	**Bindushri**		

FORENOON SESSION DETAILS







18/06/2018 C Programming

Intro

C is a general-purpose programming language that has been around nearly 50 yrs.

C has been used for everything from operating systems to complex programs.

Understanding how computer memory works is an important aspect of the C programming language.

Basic hello world program

```
#include <stdio.h>
int main(){
  printf("Hello, world!\n");
  return 0;
}
```

Data types

C supports the following basic data types;

int: integer, a whole number.

float: floating point, a number with a fractional part.

double: double-precision floating point value.

char: Single character

```
#include <stdio.h>
int main()
{
printf("int : %lu\n", sizeof(int));
printf("float: %lu\n", sizeof(float));
printf("double: %lu\n", sizeof(double));
printf("char: %lu\n", sizeof(char));
return 0;
}
```

Variables

- A variable is a name for an area in memory
name of variable also called identifier. must begin with either a letter or an underscore and can be combined of letters, digits and underscore char:

ex: int my_var;
my_var = 52;

```
#include<stdio.h>
int main(){
int a, b;
float salary = 56.07;
char letter = 'z';
a = 8;
b = 9;
int c = a+b;
printf("%d\n", c);
printf("%f\n", salary);
printf("%c\n", letter);
return 0;
```

Constants

A constant stores a value that cannot be changed from its initial assignment.

```
#include <stdio.h>
int main(){
const double PI = 3.14;
printf("%lf", PI);
return 0;
}
```

another way defining constant with #define.

```
#include <stdio.h>
#define PI 3.14
int main(){
printf("%lf", PI);
return 0;
}
```

→ INPUT & OUTPUT

C supports a number of ways for taking user input getchar() Returns the value of the next single character input.

```
#include <stdio.h>
int main(){
    char a = getchar();
    printf("you entered : %c", a);
return 0;
}
```

A string is stored in a char array.

```
#include<stdio.h>
int main()
{
char a[100];
gets(a);
printf("you entered : %s", a);
return 0;
}
```

→ scanf() scans i/p that matches format specifiers.

```
#include <stdio.h>
int main()
{
int a;
scanf("%d", &a);
printf("you entered : %d", a);
return 0;
}
```

```
#include<stdio.h>
int main()
{
int a, b;
printf("Enter two numbers: ");
scanf("%d %d", &a, &b);
printf("%d", "\n sum: %d", a+b);
return 0;
}
```

Output

Printf() → generates output

Putchar() outputs a single character.

```
#include <stdio.h>
int main()
{
 char a = getchar();
  printf("You entered: ");
  putchar(a);
  return 0;
 }
```

A string is stored in a char array

```
#include <stdio.h>
int main()
{
 char a[100];
 gets(a);
  printf("You entered: ");
 puts(a);
 return 0;
 }
```

Formatted input: The scanf() function is used to assign input to variable. ~~A call to the~~

```
int x;
float num;
 char text[20];
 scanf("%d %f %s", &x, &num, text);
```

comments

comments are explanatory information that you can include in a program to benefit the reader of your code.

operators

Arithmatic operators

C supports arithmatic operators + (addition) - (subtraction), * (multiplication), / (division), and %(modulus division)

```
#include <stdio.h>
int main(){
int length =10;
int width=5;
int area;
area = length * breadth;
printf ("%d \n", area);
return 0;
}
```

Division

```
#include<stdio.h>
int main() {
  int i1=10;
  int i2=3;
  int quotient, remainder;
  float f1=4.2;
  float f2=2.5;
  float result;
  quotient=i1/i2;
  remainder= i1%i2;
  result= f1/f2;
return 0;
```

Operator precedence

C evaluates a numeric expression based on operator precedence.

Conditionals and loops

Conditionals are used to perform different computations or actions depending on weather a condition evaluates to true or false.

the if statement is called a conditional control structure because it executes statements when expression is true

```
#include <stdio.h>
int main(){
int score=89;
if (score>75)
  printf("you passed.\n");
return 0;
}
```

Relational operators:-

< lessthan, <= lessthan or equal to, >, >=, ==, !=

if-else statement

```
#include <stdio.h>
int main(){
int score=89;
if (score>=90)
printf("Top 10%%.\n");
else
printf("Less than 90.\n");
return 0;
```

Conditional expressions

```
#include <stdio.h>
int main() {
int y;
int x=3;
y= (x>=5) ? 5 : x;
if (x>=5)
y=5;
else
y=x;
return 0;
}
```

Nested if statements

```
if (profit >1000)
if (clients >15)
  bonus =100;
else
  bonus =25;
```

The switch statement

The switch statement branches program control by matching the result of an expression with a constant case value.

```
switch (exp){
  case Val1:
    statements
  break;
  case val 2:
    statement
  break;
  default:
```

→ Logical operators

Logical operators && and || are used to form a compound Boolean expression that tests multiple conditions

the && operator.

```
if (n>0 && n<=100)
  Printf ("Range (1-100). \n");
```

→ The || operator.

```
if (n ==999 || (n>0 && n<=100
Printf ("Input valid. \n");
```

→ The ! operator

* while loop

```
while (exp) {
   Statements
}
#Include <stdio.h>
int main(){
  int count =1;
while (count < 8) {
Printf ("count =%d\n", count
count++;
} return 0;
}
```

* Do-while

```
int main(){
int count=1;
do { printf ("count =%d\n", count)
  count++;
} while (count <8);
```

Break and Continue

- the break statement was introduced for use in the switch statement

```
int num=5;
while (num >0){
  if (num==3)
  break;
  Printf("%d\n",num);
  num--;
}
```

```
#include <stdio.h>
int main(){
  int num=5;
  while (num>0){
    num--;
    if (num==3)
      continue;
    Printf("%d\n",num);
  }
```

o/p :- 4
2
1
0

*for Loop

- the for statement is a loop structure that executes statements a fixed number of times

```
for (initvalue; condition; incr){
  Statements;
}
```

```
int i;
int max=10;
for (i=0; i<max; i++){
  Printf("%d\n",i);
}
```

Functions in C

- functions are central to C programming and are used to accomplish a program solution as a series of subtasks

```
int main(){
  int x, result;
  x=5;
  result = square(x);
  Printf("%d squared is
    %d\n", x, result);
  return 0;
}
```