Set II

1. a) Name the header file for the following built-in functions:
   i) `fabs()` **math.h** ii) `isalpha()` **ctype.h** iii) `toupper()` **ctype.h**
   iv) `cos()` **math.h**

   b) Give the output of the following program:
```
6,70,70
2,70
210,282,59290
```

   c)
   i) Give the output of the program segment given above.
   4
   8
   12
   16
   20
   24
   ii) Obtain the same output by using **`do-while`** loop.
```
#include<iostream.h>
void main()
{
int n=4;
do
{
cout<<n<<endl;
n=n+4;
}while(n<=24);
getch();
  }
```

2. a) Write C++ statement or expression for the following:
   i) Increment value of a variable `count` by 1, by using only unary operator.
   ++count;
   count++;
   ii) Check that an integer variable `yr` is divisible by 4 but not divisible by 100.
   yr % 4 = = 0 && yr % 100 != 0
   iii) All possible ways of calculating square root of a floating point variable `x` and storing the result in another floating point variable `b`.
   float a, b
   b=sqrt(a)
   b=pow(a, 0.5)

   iv) Expression to check that a character variable `alpha` contains only alphabets (without using `islower()` / `isupper()` / `isalpha()` function).
   (alpha >='a' || alpha>='A') && (alpha <='z' || alpha<='Z')

   b) i) Name all the type modifiers supported by C++.
   signed, unsigned,short, long
   ii) Name the data types that support at most two type modifiers. **char and int**
   iii) With suitable example show how will you use two type modifiers with a fundamental data type?
   unsigned short int a

c) i) Define token.
C++ token is program element or a building block of a C++ program. Tokens of a C++ program can be classified as Keyword, Identifier, Constant, Operator, String and Comment **Examples are keywords like int, break and while**. Function like main() getch()

ii) Name the token that is included with the C++ compiler. Header files are needed for which kind of token?
Keywords are included in c++ complier
To use built-in identifier we need appropriate header file

iii) Name any four operators that can work with all fundamental data type (excluding **`void`**) of C++.
+ ,- ++ --

d)
```
#include<iostream.h>
void main()
{
   int a, b, lo;
   cin >> a >> b;
   if (a > b)
      lo = a;
   else
      lo = b;
   cout << lo;
}
```
i) Replace **`if-else`** statement by ternary operator.
lo= `(a > b)? a:b;`

ii) Replace **`if-else`** statement by **`switch-case`**.
```
switch(a>b)
{
   case 0:  lo=b; break;
   case 1: lo=a; break;
}
```

3. a) i) What are the two broad categories of formal parameters?
Value parameter & reference Parameter

iii) State any two differences between the two broad categories of formal parameters.

Value Parameter	**Reference Parameter**
• Copy of actual parameter	• Alias of actual parameter
• Change in value parameter does not change actual parameter	• Change in reference parameter, updates actual parameter
• Transfer of data is one way, from calling function to called function	• Transfer of data is two ways, from calling function to called function and vice-versa
• Actual parameter may either be a variable or an expression or a constant	• Actual parameter can only be a variable, it cannot be constant or expression

iv) What is an alias? Write C++ statement(s) to create an alias of a character variable.
Some times in our program we need give new name to an existing variable. When we do that, we create a Reference or an Alias of an existing variable.
Rule: DataType& NewVariableName=OldVariableName;
We use ampersand operator (&) just after the DataType. Ampersand operator (&) works as Reference operator. Statement to create a alias of character variable
char var= 'A';
char & var1=var;

b) Explain function declaration and function definition with a suitable example.

A function declaration also called function prototype is a function header (declarator) terminated by a semi-colon where a function header consists of function name, return value of a function and optional list of parameters.

Example: **double** factorial(**int**);

A function definition is the complete function, that is, header and the body. A function declaration must appear above any use of the function's name. But function's definition, when listed separately from the function's declaration, may appear anywhere outside the body or block of main() function.

Example:

```
double factorial(int n)
{
double fact=1;
for (int k=1; k<=n; k++)
fact*=k;
return fact;
}
```

c) i) Define local variable.

A variable created inside a block (block of `if-else`, block of `while` loop, block of `for` loop, block of `do-while` loop and function block) is called a Local Variable.

ii) Mention any two characteristics of a local variable.

Characteristics of Local variable:

a) Visible inside the block and block nested blow
b) Longevity is as long as the block is active
c) Default value of a Local Variable is garbage

4. Write C++ function for the following:

a) Find the sum of the digits and the product of digits of an integer passed as a parameter to a function. Display the sum and product inside the function. Return value of the function is **`void`**.

```
void sumoddeven(int n)
{
int se=0, so=0;
while (n!=0)
{
int d=n%10;
if (d%2==0)
se+=d;
else
so+=d;
n/=10;
}
cout<<"Sum of odd digits:"<<so<<endl;
cout<<"Sum of even digits:"<<se<<endl;
}
```

```
Without function
void main()
{
int n,se=0, so=0;
cout<<"Enter Number";
cin>>n;
while (n!=0)
{
int d=n%10;
if (d%2==0)
se+=d;
else
so+=d;
n/=10;  }
cout<<"Sum of odd digits:"<<so<<
endl;
cout<<"Sum  of  even  digits:"<<se
<<endl;   }
```

b) An floating point (`x`) and a integer (`n`) is passed as parameters to the function named `seqsum`(). Return value of the function is **`double`**. The function `seqsum`() finds the sum of the series given below:

$$\frac{x^2}{2!}-\frac{x^4}{4!}+\frac{x^6}{6!}-...(-1)^{n+1}\frac{x^{2n}}{(2n)!}$$

```
double seqsum(double x, int n)
{

double y=x*x ,s=0, p=1, f=1;
for (int k=1; k<=n; k++)
{
int t=2*k;
f*=t*(t-1);
s+=y/f;
y*=-(x*x);
}
return s;
}
```

```
Without function
#include<iostream.h>
void main()
{
double x;
int n;
cout<<"Enter the value of x=";cin>>x;
 cout<<"Enter the value of n=";cin>>n;
 double y=x*x ,s=0, p=1, f=1;
for (int k=1; k<=n; k++)
{
int t=2*k;
f*=t*(t-1);
s+=y/f;
y*=-(x*x);
}
cout<< s;
}
}
```

c) Check whether an integer is a Prime or a Composite number. The integer that is to be tested is passed as parameter to the function. If the values stored in the integer is less than 2 then display an error message "Neither Prime nor Composite". If the integer is Prime number then display "Prime" and display "Composite" if the integer is Composite number. Return value of the function is **`void`**.

```
    void generateprime(int n)
    {
    if(n<2)
    cout<<"Neither Prime nor Composite";
    else
    {
    for(int k=2; k<=n; k++)
    {
    int x=2, prime=1;
    while(x<n && prime==1)
    {
    if (n%x_==0)
    prime=0;
    x++;
    }
if (prime==1)
cout<<prime<<endl;
else
cout<<"Not prime";
    }
    }
    }
```

Without function

```
#include<iostream.h>
void main( )
{
    int n;
    cout<< " Enter the No:"; cin>>n;
    if(n<2)
    cout<<"Neither Prime nor Composite";
    else {
    for(int k=2; k<=n; k++)
    {
        int x=2, prime=1;
        while(x<n && prime==1)
        {
        if (n%x==0)
            prime=0;
            x++;
        }
        if (prime==1)
        cout<<prime<<endl;
        else
        cout<<"Not prime";
    }
    }
}
```

d) Generate and display all the Armstrong numbers between 1 and `n` (including 1 and `n`), where `n` is passed as a parameter to the function. Return value of the function is **`void`**. Name of the function is `genarmstrong`().

```
void genarmstrong(int n)
{
for (int k=1; k<=n; k++)
{
int sum=0, temp=k;
while (temp>0)
{
int digit=temp%10;
sum+=digit*digit*digit;
temp/=10;
}
if (sum==k)
cout<<k<<endl;
}
}
```

Without function

```
void main( )
{
int n;
  cout<<"enter Number:"; cin>>n;
  for (int k=1; k<=n; k++)
  {
    int sum=0, temp=k;
    while (temp>0)
    {  int digit=temp%10;
      sum+=digit*digit*digit;
      temp/=10;
    }
    if (sum==k)
      cout<<k<<endl;
  }
}
```