This question paper contains **20** printed pages]

Roll No. | | | | | | | | | | |

S. No. of Question Paper : **5776**

Unique Paper Code : **2342011204**

Name of the Paper : **Programming Using C++**

Name of the Course : **B.Sc. (H) Computer Science**

Semester : **II**

Duration : **3 Hours** Maximum Marks : **90**

(Write your Roll No. on the top immediately on receipt of this question paper.)

Paper has two sections. *All* the questions in Section A are compulsory.

Answer any *four* questions from Section B.

Parts of question must be answered together.

Section A

(Compulsory)

1. (*a*) Draw flowchart for finding sum of first n natural numbers. 2

(*b*) Which of the following are invalid identifiers in C++ ? 2

(*i*) `A4#1`

(*ii*) `1_Class`

(*iii*) `Class`

(*iv*) `Monthly_rate`

P.T.O.

(*c*) Write function `cube( )` which computes the cube of an integer passed as a pointer in the following program and give the output of the following code segment. 3

```
void cube(int *numptr); //prototype
int main()
    {
    int number=5;
    cout<<"The number is:"<<number<< endl;
    cube(&number);
    cout<<"The number after a call:"<< number<<endl;
    return 0;
    }
```

(*d*) Give the output that will be produced on execution of the following code segments :

(*i*) 3

```
void f(int x, int &y)
    {
      x+=10;
      y+=x;
    }
int main()
{
   int num1=12, num2=5;
   cout<<"\nBefore:"<<"num1 = "<<num1<<" , num2="<<num2;
   f(num1,num2);
   cout<<"\nAfter:"<<"num1="<<num1<< ", num2="<<num2;
   return 0;
}
```

(ii)
```
class Test
{
  private:
    static int count;
  public:
  Test & fun( );
  };
  int Test::count = 0;
  Test & Test::fun()
  {
    Test::count++;
    cout << Test::count << " ";
    return *this;
  }
 int main()
 {
    Test t;
    t.fun();
    t.fun();
    return 0;
 }
```


(iii)
```
void func(int a, int b)
{
if (a == 0 || b == 0)
{
throw "The product is zero. Provide non-zero values.\n";
}
else
{
cout<<"Product of "<<a<<" and"<<b<<" is: "<<a*b<<endl;
}
}
int main()
{
  try {
    func (5,0);
    func(2,5);
    }
catch (const char* e)
    {
      cout<<e;
    }
  return 0;
}
```

(e) Define a function omit (string W) which takes a string W as an argument and returns the word omitting all vowels from W. For example : BEAUTIFUL returns BTFL. 4

(f) Give the output of the following code segment : 5

```
class sample
{
protected:
    int count;
public:
    void initialize( )
    {
       count = 0;
       cout<< "sample with count: "<< count << endl;
    }
    void setData(int k)
    {
       count = k;
       cout<< "sample with count: "<< count << endl;
       }
};
class newsample: protected sample
{
```



```
private:
    int newcount;
public:
    void setValues(int k)
    {
    setData(k);
    newcount = count * 20;
    cout << "newsample with newcount: "<< newcount <<endl;
    cout << "newsample with count: " << count << endl;
    }
};
int main()
{
    sample S3;
    S3.initialize ();
    sample S1;
    S1. setData(5);
    newsample S2;
    S2. setValues(10);
    return 0;
}
```

(*g*) Write a function template `minimum (T arr , int size)` where `T` is the generic type of array `arr` to find minimum from the array. Use this function for finding minimum from integer, character and floating-point number array. 5

Part B

(Attempt any *four* questions)

2 (*a*) Give the output of the following code segment: 3

```
void display(char c = '*', int count= 3)
{
for(int i = 1; i <= count; ++i)
{
    cout << c;
}
cout << endl;
}
int main()
{
    int count = 5;
    cout << "First Call:";
    display();
    cout << "Second Call: ";
    display('#');
    cout << "Third Call:";
    display('$', count);
    return 0;
}
```


(*b*) Write recursive function `fact (int x)` for finding the factorial of n. Use this function to find $C(n,r)$, where $C(n,r) = \frac{n!}{r! * (n-r)!}$. 6

(*c*) Given a following code segment. Write statements to call function `display` in Line 1-3 using object `d` of class `Derived` in main function to generate the following output : 6

Output:

Derived: display()

Intermediate: display()

Base: display()

Code segment:

```
class
{
public:
    void display()
    {
        cout << "Base: display()" << endl;
    }
};
class Intermediate : public Base
{
public:
    void display()
```

```
    {
       cout << "Intermediate: display()" << endl;
    }
};
class Derived : public Intermediate
{
public:
    void display()
    {
       cout << "Derived: display()" << endl;
    }
};
int main()
{
    Derived d;
    ------------------------//Line 1
    ------------------------//Line 2
    ------------------------//Line 3
    return 0;
}
```


3. (*a*) **Give the output that will be produced on execution of the following code segment :** 6

```
int main()
{
    int a[] = {2,9,7,21,46};
    int *pa = a;
    int *pr = pa+2;
    cout<< *pa << endl;
    cout<< *pr << endl;
    cout<< *pr-- << endl;
    pr+=2;
    cout<< *pr << endl;
    pr-=2;
    cout<< *pr << endl;
    cout<< *pr + *pa << endl;
    return 0;
}
```

(*b*) **Consider the following class :** 9

```
class Ratio
{  private:
       int numerator;
       int denominator;
```

```
    public:
    Ratio();  //constructor
    Ratio(int  n,  int  d);//constructor
    Ratio(Ratio&  r);  //copy  constructor
    //Add  two  Ratios  using  this  pointer
    Ratio&  add(Ratio  r1);
};
```

Define all functions and write `main()` to compute the equation

$$\text{Result} = \frac{2}{3} + \frac{4}{5}$$

by calling these functions using appropriate class `Ratio` objects.

4. (a) **Give the output that will be produced on execution of the following code segment :** **6**

```
class  Student
{
public:
    void  displayInfo()
    {
       cout  <<  "Student  Information"  <<  endl;
    }
    void  calculateGrade()
    {
       cout  <<  "Grade  calculation"  <<  endl;
    }
```



```
};
class  ScienceStudent:  public  Student
{
public:
    void  displayInfo()
    {
       cout  <<  "Science  Student  Information"  <<  endl;
    }
    void  calculateGrade()
    {
       cout  <<  "Grade:  A"  <<  endl;
    }
};
int  main()
{
    student  students[3];
    students[0]  =  ScienceStudent();
    students[1]  =  ScienceStudent();
    student*  topper  =  &students[1];
    for  (int  i  =  0;  i  <  2;  i++)
    {
       cout<<"Student  "<<  i  +  1  <<"  Information:  "endl;
       students[i]  .displayInfo();
       students[i].calculateGrade();
       cout  <<  endl;
```

```
}
cout<< "student Topper detail"<< endl;
topper-> displayInfo();
topper-> calculateGrade();
return 0;
}
```

(b) Write a program that multiplies two 2D matrices A and B entered by the user. The program should do the following : 9

(i) Prompt the user to enter the dimensions (rows and columns) of both matrices.

(ii) Check if matrix multiplication is possible (i.e., number of columns in Matrix A equals number of rows in Matrix B).

(iii) If multiplication is possible:

- Accept the elements of both matrices.
- Multiply the matrices and store the result in a third matrix.
- Display the result.

(iv) If multiplication is not possible, display an appropriate error message.

5. (a) Give the output after execution of the following code segment. Explain which display () function is executed with a reason for each line 1-6 : 6

```
template <class T>
void display(T value)
```



```
{
    cout<<" Display 1: "<<value<<endl;
}
template <class T, class T1>
void display(T value1, T1 value2)
{
    cout<<"Display 2: "<<value1<<" & "<<value2<< endl;
}
void display(int value)
{
    cout << "Explicit Display: "<<value<<endl;
}
int main( )
{
    display(10);          //Line 1
    display('A');         //Line 2
    display(15.67);       //Line 3
    display(110,12.78);   //Line 4
    display(94,"Good");   //Line 5
    display (12,34);      //Line 6
    return 0;
}
```

(b) Write a program to demonstrate runtime polymorphism using an Employee base class and derived classes for different employee types having the following structures : 9

Base class : Employee

- Protected data member: name as string
- Parameterized constructor to initialize name
- Virtual function: calculateSalary () that displays a generic message with computed total salary and employee name.

Derived classes :

- FullTimeEmployee:

  private data members: basicSalary, bonus

  and total salary computed as (basicSalary + bonus)

- PartTimeEmployee:

  private data members : hoursWorked, hourlyRate and total salary computed as (hoursWorked * hourlyRate)

Write calculateSalary( ) for both the derived classes and main() to show runtime polymorphism by calling calculateSalary () through the pointers to appropriate objects.

P.T.O.

6. (a) Given x = 0, y = 0, z = 1, what are the values of x, y and z after executing the following code segment : 2

```
switch(x)
    {
      case 0  :  x  =  1;
                 y  =  x+y;
                 z  =  8  ;
      case 1:    x  =  4+z;
      default:   y  =  3;
                 x  =  y;
                 z  =  1;
    }
```

(b) Find and explain the error(s) from the following code segment : 6

```
class  Base
{
   private:
      int  a;
   protected:
      int  b;
   public:
      int  c;
   Base( )
     { a=10;
       b=20;
       c=30;    }
};
```

```
class  Derived:  protected  Base
{
    private:
        int d;
    protected:
        int e;
    public:
        Derived( )
            {  d=100;
            e=200; }
        void  show( )
        {
            cout  <<"b  =  "  <<  b  <<  endl;  //  Line  1
            cout  <<  "c  =  "  <<  c  <<  endl;//  Line  2
            cout  <<  "d  =  "  <<  d  <<  endl;//  Line  3
            cout  <<  "e  =  "  <<  e  <<  endl;//  Line  4
        }
};
class  MoreDerived:  public  Derived
{
    private:
        int  g;
    public:
        MoreDerived()
        {
        g=109;
```



```
        void  display( )
          {
            cout  <<  "a  =  "  <<  a  <<  endl;//  Line  5
            cout  <<  "b  =  "  <<  b  <<  endl;//  Line  6
            cout  <<  "c  =  "  <<  c  <<  endl;//  Line  7
            cout  <<  "d  =  "  <<  d  <<  endl;//  Line  8
            cout  <<  "e  =  "  <<  e  <<  endl;//  Line  9
            cout  <<  "g  =  "  <<  g  <<  endl;// Line 10
          }
};
int  main( )
    {
    Derived  d1;
    cout  <<  d1.b  <<  endl;          //  Line  11
    cout  <<  d1.c  <<  endl;          //  Line  12
    MoreDerived  md;
    md.display();
    cout  <<  md.e  <<  endl;          //  Line  13
    cout  <<  md.g  <<  endl;          //  Line  14
    return  0;
}
```

(c) Write a program to calculate the square root of a number entered by the user. The program should follow these requirements : 7

(i) Prompt the user to enter a number.

(*ii*) If the number is negative, the program should throw an exception indicating that square root of a negative number is not allowed.

(*iii*) Use a try-catch block to catch the exception and display an appropriate error message.

(*iv*) If the number is non-negative, calculate and display the square root of the number using the built-in sqrt() function.

7. (*a*) Write a program to copy a text file `A.txt` to another file `B.txt` having all words in reverse order. For example : 7

`A. txt`: This is the original file.

`B. txt`: sihT si eht lanigiro elif.

(*b*) Find error(s) and give output that will be produced after correction in the following code segment: 8

```
class  Sample
{
    int  value;
    static  int  counter;
public:
    void  setValue(int  v)
      {
      value  =  v;
      }
    static  void  countCall()
    {
```



```
      counter++;
      cout<<"Function  called  "<<counter<<"times  .  "<<endl;
      cout  <<"Value  is:  <<"value<<endl;
    }
};
int  Sample::counter;
int  main()
{
    Sample  s1,  s2;
    s1.setValue  (5);
    s2.setValue(10);
    Sample::countCall();
    s1.countCall();
    return  0;
}
```