USN :

JAWAHARLAL NEHRU NATIONAL COLLEGE OF ENGINEERING, SHIVAMOGGA

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

1st – Internal Assessment

Semester: 4-CBCS
Subject: OOC (18CS45)
Faculty: Pradeep H K and Dr. Ragavendra R J

Date:11-07-2022
Time: 10:00 AM–11:30 AM
Max Marks: 50

Q.No		Answer Any 2 Question(s)	Marks	CO	BTL
1	a	Define function overloading? WAP to add 2 parameter, 3 parameter & 4 parameter using function overloading.	**15**	CO1	L3
	b	Define inline function with example.	**10**	CO1	L3
		OR			
2.	a	Define function default arguments? WAP to add 2 parameter, 3 parameter & 4 parameter using default arguments.	**12.5**	CO1	L3
	b	Describe class & objects with an example program.	**12.5**	CO1	L3
3	a	Show how static data members are different from normal data members.	**12.5**	CO1	L3
	b	Write C++ program to do basic arithmetic operation on two complex numbers.	**12.5**	CO1	L3
		OR			
4	a	Differentiate constructor and destructors.	**12.5**	CO1	L3
	b	Write appropriate program to show objects as function arguments (use time class contain hours and minutes as data members).	**12.5**	CO1	L3
		***************			

[illegible]

QP Scrutiny Committee
ISE Dept. JNNCE

JAWAHARLAL NEHRU NATIONAL COLLEGE OF ENGINEERING, SHIVAMOGGA

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

1st – Internal Assessment Scheme

Semester: 4–CBCS Date:11–07–2022

Subject: OOC (18CS45) Time:10:00AM – 11.30AM

Faculty: Pradeep H K and Dr. Ragavendra R J Max Marks: 50

1.a Define function overloading? WAP to add 2 parameter, 3 parameter & 4 parameter using Function overloading 3

Definition – function overloading – C++ allows two or more functions to have the same name. For this, however, they must have different signatures. Signature of a function means the number, type, and sequence of formal arguments of the function. In order to distinguish amongst the functions with the same name, the compiler expects their signatures to be different. Depending upon the type of parameters that are passed to the function call, the compiler decideswhich of the available definitions will be invoked.

Prg to add 2 parameter, 3 parameter & 4 parameter using function overloading – 12 marks

```
# include <iostream.h>
int add (int a, int b)
{
    return(a+b);
}
int add (int a, int b, int c)
{
    return(a+b+c);
}
int add (int a, int b, int c, int d)
{
    return(a+b+c+d);
}
void main(void)
{
    int p,q,r,s;
    cin  >> p >> q;
    cout << "\n sum of 2 numbers:" << add(p,q);
    cin  >> p >> q >> r;
    cout << "\n sum of 3 numbers:" << add(p,q,r);
    cin  >> p >> q >> r >> s;
    cout << "\n sum of 4 numbers:" << add(p,q,r,s);
}
```

1.b Define inline function with example – 3 marks

Inline functions are used to increase the speed of execution of the executable files. C++ inserts calls to the normal functions and the inline functions in different ways in an executable. The executable program that is created after compiling the various source codes and linking them consists of a set of machine language instructions. When a program is started, the operating system loads these instructions into the computer's memory. Thus, each instruction has a particular memory address. The computer then goes through these instructions one by one. If there are any instructions to branch out or loop, the control skips over instructions and jumps backward or forward as needed. When a program reaches the function call instruction, it stores the memory address of the instruction immediately following the function call. It then jumps to the beginning of the function, whose address it finds in the function call instruction itself, executes the function code, and jumps back to the instruction whose address it had saved earlier. Obviously, an overhead is involved in making the control jump back and forth and storing the address of the instruction to which the control should jump after the function terminates.

The C++ inline function provides a solution to this problem. An inline function is a function whose compiled code is 'in line' with the rest of the program. That is, the compiler replaces the function call with the corresponding function code. With inline code, the program does not have to jump to another location to execute the code and then jump back. Inline functions, thus, run a little faster than regular functions.

```
/*Beginning of inline.cpp*/                                  – 7 marks
# include <iostream.h>
inline double cube(double x) { return x*x*x; }
void main()
{
    double a,b;
    double c=13.0;
    a=cube(5.0);
    b=cube(4.5+7.5);
    cout << a << endl;
    cout << b << endl;
    cout << cube(c++) << endl;
    cout << c << endl;
}
```

2.a Define function default arguments? WAP to add 2 parameter, 3 parameter & 4 parameter using default arguments

Default arguments: It is possible to specify default values for some or all of the formal arguments of a function. If no value is passed for an argument when the function is called, the default value specified for it is passed. – 2.5 marks

```
# include <iostream>                                        – 10 marks
int add (int a, int b, int c=0, int d=0)
{
    return(a+b+c+d);
}
void main(void)
{
    int p,q,r,s;
    cin >> p >> q;
    cout << "\n sum of 2 numbers:" << add(p,q);
    cin >> p >> q >> r;
    cout << "\n sum of 3 numbers:" << add(p,q,r);
    cin >> p >> q >> r >> s;
    cout << "\n sum of 4 numbers:" << add(p,q,r,s);
}
```

2.b Describe class & objects with an example program – 12.5 marks

Class: Classes are to C++ what structures are to C. Both provide the library programmer a means to create new data types.

Objects: Variables of classes are known as objects. An object of a class occupies the same amount of memory as a variable of a structure that has the same data members. – 2.5 marks

– 10 marks

```
# include <iostream>
class add
{
    int a,b,c;
public:
    void getdata(void)
    {
        cin >> a >> b;
    }
    void sum(void)
    {
        c=a+b;
    }
    void putdata(void)
    {
        cout << "\n a :" << a
        "\n b:" << b
        "\n c:" << c;
    }
};
void main(void)
{
    add x;
    x.getdata();
    x.sum();
    s.putdata();
}
```

3.a Show how static data members are different from normal data members

– 12.5 marks

```
# include <iostream>
# include <math.h>
```

```
class item
{
      static int count;
      int number;
      public:
      void getdata(int x)
     {
         number=x;
         count++;
     }
     void putdata(void)
     {
         cout << "\n count :" << count;
     }
};

int  item::count=10;   // Explicit allocation

void main(void)
{
     item a,b,c;
     a.putdata();
     b.putdata();
     c.putdata();

     cout << " \n after the data";

     a.getdata(100);
     b.getdata(200);
     c.getdata(300);
     a.putdata();
     b.putdata();
     c.putdata();
}
```

3.b WAP to add 2 complex numbers – 12.5 marks

```
# include <iostream>
# include <math.h>
class complex {
     float r, im;
public:
     complex() { } //default constructor
     complex(float h) { r = im = h; }
     complex(float h, float m) { r = h; im = m; }

void show(void) {
         if ( im > 0 ) cout << r << "+i" << im;
            else
         cout << r << " -i" << abs(im);
     }
     friend complex sum(complex,complex);
};
```

```
complex sum(complex c1, complex c2)
{
	complex c3;
	c3.r=c1.r+c2.r;
	c3.im=c1.im+c2.im;
	return(c3);
}
void main(void)
{
	complex A(5,35),B(7,-43),C;
	C=sum(A,B);
	cout << "\n A :"; A.show();
	cout << "\n B :"; B.show();
	cout << "\n C :"; C.show();
	complex c4,c5,c6;
	float r1,im1;
	cout << "\n enter value for r and im";
	cin >> r1 >> im1;
	c4=complex(r1,im1);
	cout << "\n enter value for r and im";
	cin >> r1 >> im1;
	c5=complex(r1,im1);
	c6=sum(c4,c5);
	cout << "\n A :"; c4.show();
	cout << "\n B :"; c5.show();
	cout << "\n C:"; c6.show();
}
```

4.a Differentiate constructor and destructors – 12.5 marks

The constructor gets called automatically for each object that has just got created. It appears as member function of each class, whether it is define or not. It has the same name as that of the class. It may or may not take parameters. It does not return anything (not even void). The prototype of a constructor is

<class name> (<parameter list>);

The destructor gets called for each object that is about to go out of scope. It appears as a member function of each class whether we define it or not. It has the same name as that of the class but prefixed with a tilde sign. It does not take parameters. It does not return anything (not even void). The prototype of a destructor is

~<class name> (); – 2.5 marks

– 10 marks

```
# include <iostream>
# include <math.h>
int count=0;
class alpha
{
public:
	alpha()
	{
		count++;
		cout << "\n object created " << count;
	}
```

```
    ~alpha()
    {
        cout << "\n object destroyed " << count;
        count--;
    }
};
void main(void)
{
    alpha A1, A2, A3,A4;
    cout << "\n main block";
    {
        cout << "\n enter Block 1";
        alpha A5;
    }
    {
        cout << "\n enter Block 2";
        alpha A6;
    }
}
```

4.b Write appropriate program to show objects as function arguments (use time class contain hours and minutes as data members) – 12.5 marks

```
# include <iostream>
# include <math.h>
class time
{
    int hours, minutes;
    public:
    void gettime(int h, int m)
    {
        hours=h;
        minutes=m;
    }
    void puttime(void)
    {
        cout << "\n Time: >> " << hours << ":" << minutes;
    }
    void sum(time t1, time t2)
    {
        hours=t1.minutes+t2.minutes;
        minutes=hours%60;
        hours=hours/60;
        hours=hours+t1.hours+t2.hours;
    }
};
void main(void)
{
    time t1,t2,t3;
    t1.gettime(5,35);
    t2.gettime(7,43);
    t1.puttime();
    t2.puttime();
    t3.sum(t1,t2);
    t3.puttime();
}
```