Techno India University

Name: Rohan Ghosh

Batch: BCS2B

Id: 181001001122

Subject: Object Oriented Programming

GROUP – A

Choose the correct one:-

1) What is the range of int data type?

Ans: d) None of these

2) Which of the following function initializes variables in a class:

Ans: a) Constructor

3. Indicate which data type is not part of standard C++.

Ans: c) real

4) Which is true regarding inline function?

Ans: b) It saves memory space

5) Friend function

Ans: b) Can not have the pointer

GROUP - B

1) Differentiate between call by value and call by reference with reference.

ANSWER:

CALL BY VALUE	CALL BY REFERENCE
Calling function sends copies of data of actual parameter to the caller function’s formal parameter.	Here calling function sends address of actual parameter to the caller function.
The formal parameters are normal variable.	The formal parameters are pointer variable.
Changes in formal parameter doesn’t affect actual parameter.	Changes in formal parameter affectactual parameter.
Eg:	Eg:

```
#include <stdio.h>

class ByValue{

    void swapByValue(int a, int b){

        int t;

        t = a; a = b; b = t;

    }

    int main(){

        int n1 = 10, n2 = 20;

        swapByValue(n1, n2);

        printf("n1: %d, n2: %d\n", n1, n2);
```

```
#include <stdio.h>

class ByReference{

    void swapByReference(int *a, int
*b){

        int t;

        t = *a; *a = *b; *b = t;

    }

    int main(){

        int n1 = 10, n2 = 20;

        swapByReference(&n1, &n2);

        printf("n1: %d, n2: %d\n", n1,
```

}
 	
 }
 	
 OUTPUT:n1: 10, n2: 20
	n2);
	
	 }
	
	 }
	
	 Output:n1: 20, n2: 10

2) Discuss difference between C and C++.

ANSWER:

C	C++
C was developed in 1969 at AT& T Bell labs by Dennis Ritchie.	C++ was developed in 1979 by Bjame Stroustrup.
C is not an object oriented programming since, it does no support polymorphism, encapsulation, and inheritance.	Whereas C++is a object oriented programming as it does support polymorphism, encapsulation, and inheritance.
C contains 32 keywords.	C++ contains 52 keywords.
Data and functions are separated in C because it is a procedural programming language.	While in c++ data and functions are encapsulated together in form of an object.
Virtual and friend functions are not supported by C.	Virtual and friend functions are supported by C++.

3) How to specify default argument in a function?

ANSWER:

In C++ programming, we can provide default values for function parameters. The idea behind default argument is simple. If a function is called by passing argument/s, those arguments are used by the function.

But if the argument/s are not passed while invoking a function then, the default values are used.Default value/s are passed to argument/s in the function prototype.

```
// C++ Program to demonstrate working of default argument

#include <iostream>

using namespace std;

void display(char = '*', int = 1);

int main(){

   cout << "No argument passed:\n";

   display();

   cout << "\nFirst argument passed:\n";

   display('#');

   cout << "\nBoth argument passed:\n";

   display('$', 5);

   return 0;

}
```

```
void display(char c, int n){
    for(int i = 1; i <= n; ++i){
        cout << c;
    }
    cout << endl;
}
```

Output-

```
No argument passed:
*
First argument passed:
#
Both argument passed:
$$$$$
```

Explanation-In the above program, we can see the default value assigned to the arguments void display(char = '*', int = 1);.At first, display() function is called without passing any arguments. In this case, display() function used both default arguments c = * and n = 1.

Then, only the first argument is passed using the function second time. In this case, function does not use first default value passed. It uses the actual parameter passed as the first argument c = # and takes default value n = 1 as its second argument.

When display() is invoked for the third time passing both arguments, default arguments are not used. So, the value of c = $ and n = 5.

4) List the rule of defining a constructor?What is the return type of constructor?

ANSWER:

Rules of defining a constructor-

1. Constructor should be same name as the class and declared in the public section of the class.

2. Constructors are invoked automatically whenever the object is created.

3. Constructors do not have return type.

4. Constructor can not be inherited, but from the derived class we can call the base class constructors.

5. Constructor can not be virtual.

6. It is not possible to refers to the address of constructors.

7. It is not possible to use the constructor as member of union if the object is created with constructor.

Return type of constructors-Constructors have no return type, not even void. This is because the implicit return type of a class' constructor is the class type itself.

5) What are the advantages of using new rather than malloc()?

Answer:

Some of the advantages of using new over malloc are listed below:-

1. new does not need the sizeof() operator where as malloc() needs to know the size before memory allocation.
2. Operator new can make a call to a constructor where as malloc() cannot.
3. new can be overloaded malloc() can never be overloaded.
4. new could initialise object while allocating memory to it where as malloc () cannot.

6) Explain why using const is a better idea than an equivalent #defines?

Answer:

The advantage of const over #define is type checking. Since const have exactly the same properties as a normal variable we can have pointers to const variables, we can pass them around, typecast them and any other possibility that can be done with a normal variable.

7) Difference between stack memory and heap memory?

Answer:

Stack memory	**Heap memory**
Memory is allocated in a contiguous order	Memory is allocated in random order.
Allocation and deallocation is automatic by compiler	Allocation and deallocation is done by programmer
Cost is less	Cost is high
This memory can be accessed fast and in low time	Access time of this memory is high
Fixed memory size	Resizing is possible

8) What is memory leak?

Answer:

Memory Leak occurs when a memory is created in heap and is never freed. Suppose if a memory is allocated using new keyword and we don't use the delete() function or delete [] opeartor after the use of that allocated memory is over then that phenomenon is called memory leak.

Memory leak is more dangerous in case of daemons or server as they have a limited memory usage bar.

9) Is it necessary that a constructor in a class should always be public?

Answer:

No it is not necessary for a constructor to always be public. But in most cases it is kept public so that any other classes can instantiate values. It can be made private if we don't want other class to be able to create an instance.

10) Explain Singleton class in C++.

ANSWER:

Singleton design pattern is a software design principle that is used to restrict the instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system. For example, if you are using a logger, that writes logs to a file, you can use a singleton class to create such a logger. You can create a singleton class using the following code −

Example

```
#include <iostream>
using namespace std;

class Singleton {
   static Singleton *instance;
   int data;
   // Private constructor so that no objects can be created.
```

```
  Singleton() {
    data = 0;
  }
  public:
  static Singleton *getInstance() {
    if (!instance)
    instance = new Singleton;
    return instance;
  }
  int getData() {
    return this -> data;
  }
  void setData(int data) {
    this -> data = data;
  }
};

//Initialize pointer to zero so that it can be initialized in first
call to getInstance
Singleton *Singleton::instance = 0;

int main(){
  Singleton *s = s->getInstance();
  cout << s->getData() << endl;
  s->setData(100);
  cout << s->getData() << endl;
  return 0;
}
```

Output:

```
0
100
```

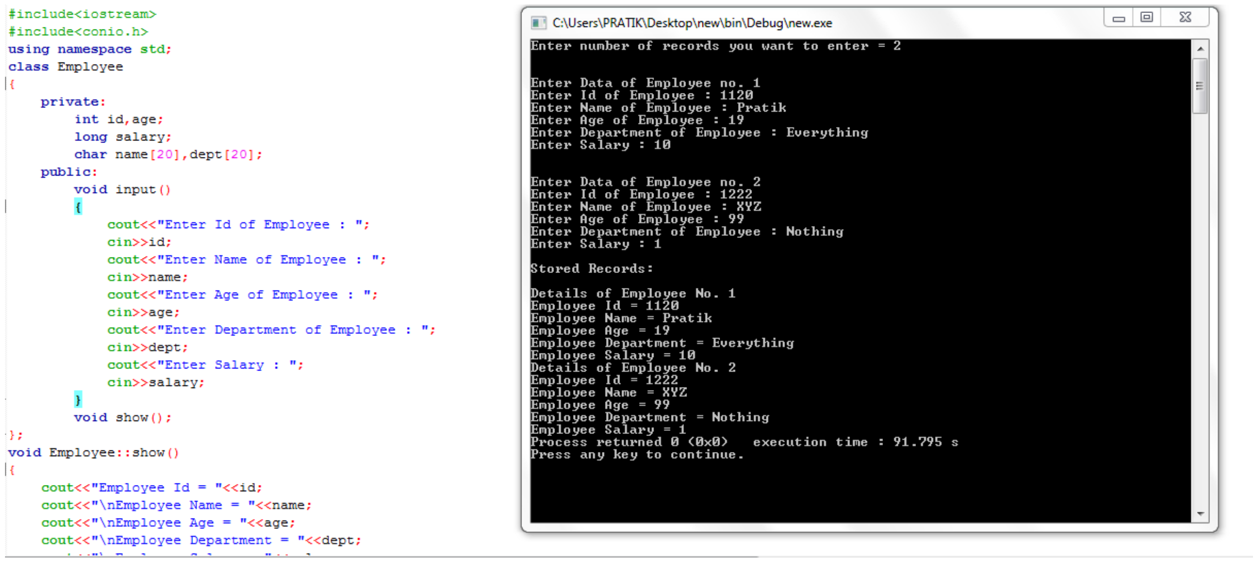
GROUP – C

1.Create a Employee Directory based on Employee Class . The class must have the following variables – Id , Name , Dept ,Salary , Age . Use dynamic memory allocation for creating an array of objects to achieve the aforementioned purpose.

```
#include<iostream>
#include<conio.h>
using namespace std;
class Employee
```

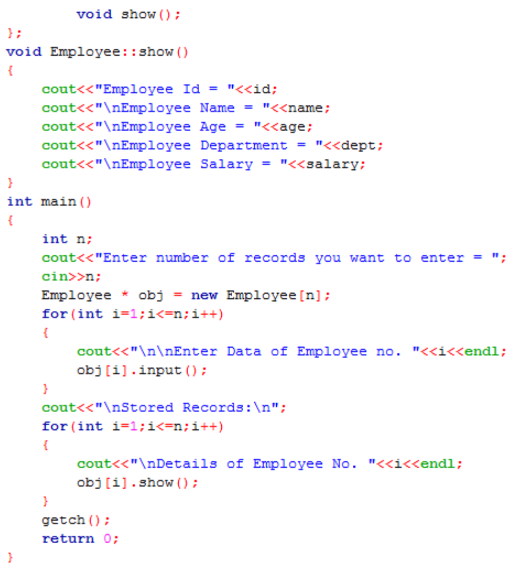
```
{
   private:
      int id,age;
      long salary;
      char name[20],dept[20];
   public:
      void input()
      {
         cout<<"Enter Id of Employee : ";
         cin>>id;
         cout<<"Enter Name of Employee : ";
         cin>>name;
         cout<<"Enter Age of Employee : ";
         cin>>age;
         cout<<"Enter Department of Employee : ";
         cin>>dept;
         cout<<"Enter Salary : ";
         cin>>salary;
      }
      void show();
};
void Employee::show()
{
   cout<<"Employee Id = "<<id;
   cout<<"\nEmployee Name = "<<name;
   cout<<"\nEmployee Age = "<<age;
   cout<<"\nEmployee Department = "<<dept;
   cout<<"\nEmployee Salary = "<<salary;
}
int main()
{
   int n;
   cout<<"Enter number of records you want to enter = ";
   cin>>n;
   Employee * obj = new Employee[n];
   for(int i=1;i<=n;i++)
   {
      cout<<"\n\nEnter Data of Employee no. "<<i<<endl;
      obj[i].input();
```

```
    }
    cout<<"\nStored Records:\n";
    for(int i=1;i<=n;i++)
    {
        cout<<"\nDetails of Employee No. "<<i<<endl;
        obj[i].show();
    }
    getch();
    return 0;
}
```



```
#include<iostream>
#include<conio.h>
using namespace std;
class Employee
{
    private:
        int id,age;
        long salary;
        char name[20],dept[20];
    public:
        void input()
        {
            cout<<"Enter Id of Employee : ";
            cin>>id;
            cout<<"Enter Name of Employee : ";
            cin>>name;
            cout<<"Enter Age of Employee : ";
            cin>>age;
            cout<<"Enter Department of Employee : ";
            cin>>dept;
            cout<<"Enter Salary : ";
            cin>>salary;
        }
        void show();
};
void Employee::show()
{
    cout<<"Employee Id = "<<id;
    cout<<"\nEmployee Name = "<<name;
    cout<<"\nEmployee Age = "<<age;
    cout<<"\nEmployee Department = "<<dept;
```

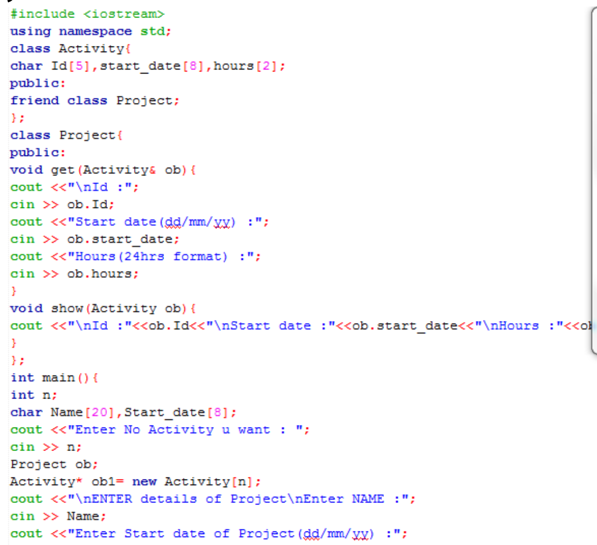
```
Enter number of records you want to enter = 2


Enter Data of Employee no. 1
Enter Id of Employee : 1120
Enter Name of Employee : Pratik
Enter Age of Employee : 19
Enter Department of Employee : Everything
Enter Salary : 10


Enter Data of Employee no. 2
Enter Id of Employee : 1222
Enter Name of Employee : XYZ
Enter Age of Employee : 99
Enter Department of Employee : Nothing
Enter Salary : 1

Stored Records:

Details of Employee No. 1
Employee Id = 1120
Employee Name = Pratik
Employee Age = 19
Employee Department = Everything
Employee Salary = 10
Details of Employee No. 2
Employee Id = 1222
Employee Name = XYZ
Employee Age = 99
Employee Department = Nothing
Employee Salary = 1
Process returned 0 (0x0)   execution time : 91.795 s
Press any key to continue.
```



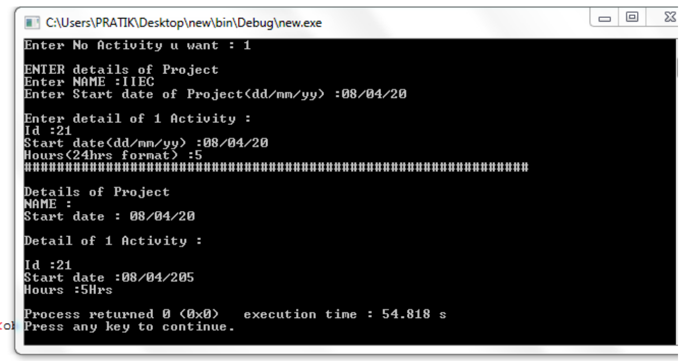
```
        void show();
};
void Employee::show()
{
    cout<<"Employee Id = "<<id;
    cout<<"\nEmployee Name = "<<name;
    cout<<"\nEmployee Age = "<<age;
    cout<<"\nEmployee Department = "<<dept;
    cout<<"\nEmployee Salary = "<<salary;
}
int main()
{
    int n;
    cout<<"Enter number of records you want to enter = ";
    cin>>n;
    Employee * obj = new Employee[n];
    for(int i=1;i<=n;i++)
    {
        cout<<"\n\nEnter Data of Employee no. "<<i<<endl;
        obj[i].input();
    }
    cout<<"\nStored Records:\n";
    for(int i=1;i<=n;i++)
    {
        cout<<"\nDetails of Employee No. "<<i<<endl;
        obj[i].show();
    }
    getch();
    return 0;
}
```

2. Suppose a company executes multiple projects . Each project must have one or more activities . Project class contains 2 attributes : Name, Start date, Array of Activity objects. Activity class contains 3 attributes : Id, start date, hours. Write a program to demonstrate this.

```
#include <iostream>
using namespace std;
class Activity{
  char Id[5],start_date[8],hours[2];
  public:
  friend class Project;
};
class Project{
  public:
  void get(Activity& ob){
    cout <<"\nId :";
    cin >> ob.Id;
    cout <<"Start date(dd/mm/yy) :";
    cin >> ob.start_date;
    cout <<"Hours(24hrs format) :";
    cin >> ob.hours;
  }
  void show(Activity ob){
    cout <<"\nId :"<<ob.Id<<"\nStart date :"<<ob.start_date<<"\nHours
 :"<<ob.hours<<"Hrs"<<endl;
  }
};
int main(){
  int n;
  char Name[20],Start_date[8];
  cout <<"Enter No Activity u want : ";
  cin >> n;
  Project ob;
  Activity* ob1= new Activity[n];
  cout <<"\nENTER details of Project\nEnter NAME :";
  cin >> Name;
  cout <<"Enter Start date of Project(dd/mm/yy) :";
  cin >> Start_date;
  for(int i=0;i<n;i++){
    cout<<"\nEnter detail of "<<i+1<<" Activity :";
    ob.get(ob1[i]);
  }
  cout <<"\n\nDetails of Project\nNAME : "<<Name<<"\nStart date :
"<<Start_date<<endl;
  for(int i=0;i<n;i++){
    cout<<"\nDetail of "<<i+1<<" Activity :"<<endl;
    ob.show(ob1[i]);
  }
  return 0;
}
```



```
#include <iostream>
using namespace std;
class Activity{
char Id[5],start_date[8],hours[2];
public:
friend class Project;
};
class Project{
public:
void get(Activity& ob){
cout <<"\nId :";
cin >> ob.Id;
cout <<"Start date(dd/mm/yy) :";
cin >> ob.start_date;
cout <<"Hours(24hrs format) :";
cin >> ob.hours;
}
void show(Activity ob){
cout <<"\nId :"<<ob.Id<<"\nStart date :"<<ob.start_date<<"\nHours :"<<ob
}
};
int main(){
int n;
char Name[20],Start_date[8];
cout <<"Enter No Activity u want : ";
cin >> n;
Project ob;
Activity* ob1= new Activity[n];
cout <<"\nENTER details of Project\nEnter NAME :";
cin >> Name;
cout <<"Enter Start date of Project(dd/mm/yy) :";
```



C:\Users\PRATIK\Desktop\new\bin\Debug\new.exe

```
Enter No Activity u want : 1

ENTER details of Project
Enter NAME :IIEC
Enter Start date of Project(dd/mm/yy) :08/04/20

Enter detail of 1 Activity :
Id :21
Start date(dd/mm/yy) :08/04/20
Hours(24hrs format) :5
##############################################################

Details of Project
NAME :
Start date : 08/04/20

Detail of 1 Activity :

Id :21
Start date :08/04/205
Hours :5Hrs

Process returned 0 (0x0)   execution time : 54.818 s
Press any key to continue.
```