Name :- Sahil Ashraf

DATE : __/__/__
PAGE : ______

Choose the correct option

(1) Stack

(2) Compiler Error in line " Derived * dp = new Base; "

(3) Inaccessible.

(4) Programmer have to always call destructor at the end of the program.

(5) True.

DATE :__/__/__
PAGE :______

SHORT ANSWER TYPE QUESTIONS

1. The new operator requests for the memory allocation in heap. If the sufficient memory is available, it intializes the memory to the pointer variable & returns its address.

e.g.

```
 # include <iostream>
using namespace std;
int main(){
int * ptr 1 = NULL;
ptr 1 = new int;
float * ptr 2 = new float(223.324);
int * ptr 3 = new int[28];
* ptr 1 = 28
cout << "Value of pointer variable 1: "<< *ptr1 <<
endl;
cout  "    "    "    "    "    2:" <<* ptr 2<<
endl;
if (! ptr 3)
cout << "Allocation of memory failed \n";
else {
for(int i = 10; i< 15; i++)
  ptr 3[i] = i+1;
cout << "Value of store in block of memory:";
for( int i = 10; i<15; i++)
 cout << ptr3 [i] <<" ";
 }
          delete ptr 1;
          delete ptr 2;
          delete ~~ptr~~ [] ptr 3;
     return 0;
     }
```

DATE : __/__/__
PAGE : ______

Output :
```
Value of pointer variable 1 : 28
  "    "    "       "     2 : 299.121
  "    "  store in block of memory : 11 .
```

The delete operator is used to deallocate the memory. User has privilege to dellocate the created pointer variable by this delete operator

(3) Procedural programming can be defined as a programming model which is derived from structured programming based upon the concept of calling procedure. Procedures also known as routines, subroutines or functions, simply consist of a series of computational steps to be carried out.

Object oriented programming can be defined as a programming model which is based upon the concept of projects. Objects contain data in the form of attributes and code in the form of methods .

Scanned with CamScanner

LONG ANSWER TYPE QUESTIONS

(A) Polymorphism is a Greek word that means to have many forms. It occurs when you have a heirarchy of classes related through inheritance.
Polymorphism causes a member function to behave differently based on the object that calls/invokes it. though we have one function, it behaves differently under different circumstances e.g suppose we have the function makeSound(). When a cat calls this function it will produce the meow sound. when a cow invokes the same function, it will provide the moow sound.

⇒ Following are the types of polymorphism :-
① Compile time polymorphism :-
It is achieved through function overloading and operator overloading.
We invoke the overloaded functions by matching the number and type of arguments. The information is present during compile-time.

Function overloading :- It occurs when we have many functions with similar names but different arguments. The arguments may differ in terms of number or type.
e.g.

```
# include <iostream>
 using namespace std;
 void test (int i) {
cout << "The int is " << i << endl;
}
void test (double f) {
```

```
cout<<"The float is" <<f<<endl
}
void Test (char const *ch){
cout <<"The char* is" << ch<<endl;
}
int main (){
Test (5);
Test ( 5.5);
test ("five");
return 0;
}
```

<u>Output:</u>

```
        The int is 5
       The float is 5.5
     The char* is five
```

We have three functions with the same name but different types of arguments. We have achieved polymorphism.

<u>Operator Overloading:-</u>

In this type we define a new meaning for a C++ operator. It also changes how the operator works. e.g we can define the + operator to concatenate two strings. We know it as the addition operator for adding numerical values.

<u>e.g.</u>

```
    # include < iostream >
    using namespace std;
  class ComplexNum{
   private :
```

```
int real, over;
public:
  ComplexNum( int rl = 0, int ov = 0 {
 real = rl;
  over = ov;
  }
ComplexNum operator+ (Complex Num const &obj){
Complex Num result;
result. real = real + obj. real;
result. over = over+ obj. over
return result;
 }.
void print () {
cout << real << "+ i" << over << endl;
 }
};
 int main ()
{
 ComplexNum c1(10, 2) , c2(3,7);
ComplexNum c3 = c1+c2;
 c3. print ();
 }
```

output = 13+i9

(2) Runtime Polymorphism :-
It happens when an object's method is invoked/called during runtime rather than during compile time. It is achieved through function over-riding

Function overriding :-
It occurs when a function of the base class is given a new definition in a derived class. At that time, we can say the fn. has

Overriden

e.g.

```
    # include <iostream>
    using namespace std;
    class Mammal {
  public:
     void eat () {
cout << "Mammals eat....." ;
     }
   };
class Cow: public Mammal {
public:
   void eat () {
cout << "Cows eat grass......" ;
  }
   };
int main (void) {
Cow c = Cow();
c.eat ();
return 0;
}
```

Output :- Cows eat grass.....

DATE / /
PAGE

(B)
```
#include<bits/stdc++.h>
using namespace std;
void sort012 (int a[], int arr-size)
{
  int lo = 0;
  int hi = arr_size -1;
  int mid =0;

while( mid<=hi) {
 switch (a[mid]) {
case 0:
swap(a[lo++), a[mid++]);
 break;

case 1:
 mid++;
 break;
case 2:
swap(a[mid], a[hi--]);
 break;
}
 }
  }
void printArray(int arr[ ], int arr-size)
{
 for(int i=0; i< arr_size; i++)
cout << arr[i]<<" ";
} int main()
{
 int arr[]= {1,1,2,2,0,0,2,1,2};
```

Scanned with CamScanner

DATE : __/__/__
PAGE : ____

```
int n = size of (arr) / size of (arr[0]);
sort ~~0~~     sort012(arr, n);
cout <<" array after segregation";
printArray(arr, n);
 return 0;
}
```

DATE :__/__/__
PAGE :____

(c)
```
#include <iostream >
using namespace std;
class member
{
 char name [30];
 int age;
long phone_number;
char address [70];
float salary;
public:
 void printSalary()
{
 cout<<"|n salary: " <<salary;
 }
void print ()
{
 cout <<"|n name :"<<name;
 cout <<"|n age : " << age;
cout << "|n phone number: "<< phone_number;
cout <<"|n address: " << address;
 }
void input ()
{
 cout << "|n enter name: ";
 cin >> name;
 cout << "|n enter age: ";
 cin>>age;
cout <<"|n enter phone number: ";
 cin >> phone_number;
 cout <<"|n enter address: ";
  cin >> address;
```

```
cout<<"\n enter salary:";
 cin>>salary;
 }
 };
class employee : public member
{
char specialization[30];
public:
void input_emp()
{
cout <<"\n\t ----enter employee details ----"
number :: input()
cout <<"\n enter specialization :";
cin>>specialization;
}
void print_emp()
{
cout <<"\n \t ----displaying details ----";
number:: print();
cout <<"\n specialization in :"<< specialization;
member:: printSalary();
 }
};
class manager : public member
{
char department [30];
public:
void input_mgr()
{
cout <<"\n\t --- Enter manager details ---";
member :: input();
```

Scanned with CamScanner

```
cout << "\n enter department :";
 cin >> department;
}
 void print_mgr()
{
 cout << "\n\t --- displaying manager details---";
member :: print();
cout << "\n department : " << department;
member :: printSalary();
}
 };
int main()
{
 employee e;
manager m;
e. input_emp();
m. input_mgr();
e. print_emp();
m. print_mgr();
return 0;
}
```