Homework Questions of 1st April 2023

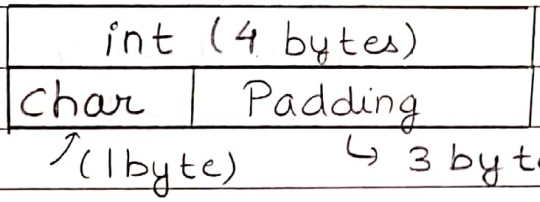
Q1 What is padding & greedy alignment in classes?

It is important to note that only the non-static data members will contribute to the size of class. The static data members & the member functions does not contribute to the size of class.

```
class Animal {
    int a;
    char b;
    public:
        void setA (int a) {
            this -> a = a;
        }
};
main() {
    cout << sizeof (Animal);
}
```

The output of the above code will be 8 bytes but shouldn't it be 4+1 = 5 bytes. The answer is no & the reason behind this is known as padding & alignment.

int → 4 bytes
char → 1 byte



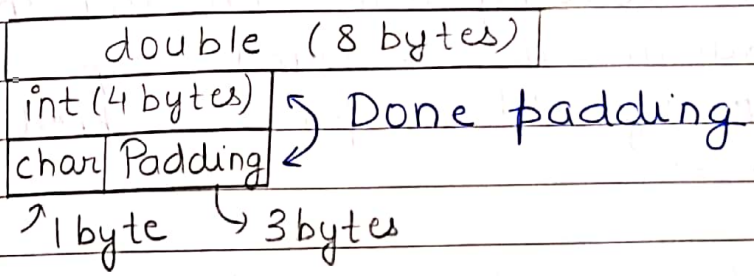
Size of class = Size of all non-static data members + padding.
Hence sizeof (Animal) = 4+1+3 = 8 bytes.

```
class Fish {
    double a;
    int b;
    char ch;
};
```

Now what is the size of Fish? The answer is ___ bytes. Let's see how.

int → 4 bytes
char → 1 byte + 3 padding bytes
double → 8 bytes

4 + 4 + 8 = 16 bytes.



The method of padding is however compiler dependent & kind of greedy. It aligns till the boundary of maximum memory allocated.

Q2 What is memory leak? How to prevent it?

Memory leakage occurs in C++ when the programmer has allocated memory with the help of new keyword & then forget to delete the memory allocated with the delete keyword.

Memory leak occurs when we use the wrong delete operator. The delete operator should be used to free a single allocated memory space whereas delete[] is used free an array of data values.

1) Always deallocate the memory after use.
2) We should have few new/delete calls in program level - ideally none.

Q3 What is garbage collector?

Garbage collection is an automated process of deleting code that is no longer needed or used. This automatically frees up the memory space.

The garbage collector considers unreachable objects garbage & releases the memory that is allocated by them. During the collection, it examines the managed heap, looking for blocks of address space occupied by unreachable objects.

Q4 What is const keyword in C++?

We use const keyword to define a constant value that can not be changed during the execution.

1) const variable → defines variable values that can never be changed.
Syntax: data type (int) ↦ value to be assigned here itself.
const int a = 5;
↳ keyword ↳ variable name
Now if we try to do any modifications to a, we would get an error.

2) constant pointer → we can't change the address of const pointer after its initialization which means it points to the same memory location.
Syntax: keyword ↗ variable name
int* const ptr = &x;
↳ datatype ↳ name of pointer

3) Pointer to constant variable → It means that pointer points to the value of const variable that can not be changed.
Syntax
int x = 7;
const int *ptr = &x;

Now x becomes the constant variable & its value can not be changed.

4) constant function arguements → If the value of function arguement is constant, then function can't change its value
Syntax
int Test (const int x) { ... }
↳ Test can't change value of x.

→ Read only function

5) const member function of class → A const member function of class never changes the value of any class data members & also never calls any non-const function.

Syntax

```
class Animal {
    public :                    → const keyword added
    void sleep() const {
        cout << "Sleeping" ;
    }
};
```

6) const data members of class → The const data members can not be assigned the values during its declaration, however they can be assigned value via constructor.

Syntax

```
class Animal {
const int x ;
public :                    x
    Animal (int y) :  (y)  {     }
};
```

7) constant objects → The value of data members can never change till the life of the object in the program. They are also known as read-only objects.

Syntax ↱ class name

```
const Animal obj ;
```

↳ keyword ↳ object name

Q5 What is static keyword in C++?

1) Static variables → When a variable is declared as static, the space for it gets allocated for the lifetime of the program. Even if function is called multiple times, space for static variable is allocated only once & the value of variable in previous call gets carried through next function call.

Syntax → datatype
static int i;
↳ keyword ↳ name

Ex→
```
void demo () {
    static int cnt; //By default cnt = 0.
    cout << cnt << " ";
    cnt ++;
}
main () {
    for (int i = 0; i < 5; i++) {
        demo ();
    }
}
```
o/p → 0 1 2 3 4

If we didn't use static keyword, then o/p will be 0 0 0 0 0.

2) Static member functions → They are allowed to access only static data members or static member functions of the class.

Syntax
Inside class ⇒ Static void sleep () { -- }

Q6 What is initializer list in C++?

Initializer list is used to initialize the data members of the class.

```
class Point {
    private :
        int x;
        int y;
    public :
        Point (int i, int j) :x(i) , y(j) { }
                              Initializer list
};
```