Lab 01 : Answers

```
#ifndef ARRAY_H
#define ARRAY_H

class Array
{
   public:
      Array();                    //Default Constructor for the class
      Array(int);                 //Copy Constructor with size
      ~Array();                   //Destructor for the class

      int sizeOfArry;            //Array size
      int *arrData;              //integer Array


      int getSize();                  //Function to access the size
      void displayArray();            //Display the content of the array
      void addToTop(int);             //Function to Add an element at the beginning
      void addToEnd(int);             //Function to Add an element at the end
      void removeFromTop();           //Function to Remove an element at the beginning
      void removeFromBottom(); //Function to Remove an element at the end
      void reverseArray();           //Function to Inverse the order of the elements in the array
      int returnSum();               //Function to Return the sum of the elements in the array
      int* checkOdd();      //Function to Return an array that contains the odd numbers only
      void writeToFile(char *); //Function to Display the content of the array in a file
      int* operator+(const Array&) const;
                                  //concatenate 2 arrays of size n and m into a new array
   protected:
   private:
};
#endif // ARRAY_H
```

```
#include "Array.h"

using namespace std;

Array::Array(){}                  //Default Constructor for the class

Array::Array(int value)           //Copy Constructor with size
{
     sizeOfArry = value;          //User Pass size
     arrData = new int[sizeOfArry];

   for(int i =0;i<sizeOfArry;i++)
      arrData[i]=i;
}

Array::~Array(){    }             // Destructor for the class
```

Lab 01 : Answers

Function to access the size

```
int Array::getSize()
{
  return sizeOfArry;
}
```

```
Array A(20);

cout<<"Size  : "<< A.getSize()<<endl;
```

Display the content of the array

```
void Array::displayArray()
{
   for(int i=0;i<sizeOfArry;i++)
       cout<<arrData[i]<<endl;
}
```

```
Array A(20);

A.displayArray();
```

Function to Add an element at the beginning

```
void Array::addToTop(int value)
{
   sizeOfArry =sizeOfArry+1;
  // with allocate memory
  arrData =
(int*)realloc(arrData,sizeof(int)*sizeOfArry);

  for(int i =sizeOfArry-1;i>0;i--)
   arrData[i] = arrData[i-1];
   arrData[0]=value;
}
```

```
Array A(20);

A.addToTop(30);
```

Function to Add an element at the end

```
void Array::addToEnd(int value)
{
   sizeOfArry =sizeOfArry+1;
   // with allocate memory
   arrData =
(int*)realloc(arrData,sizeof(int)*sizeOfArry);
   arrData[sizeOfArry-1]=value;
}
```

```
Array A(20);

A.addToEnd(50);
```

Function to Remove an element at the beginning

```
void Array::removeFromTop()
{  for(int i =0;i<sizeOfArry;i++)
   arrData[i] = arrData[i+1];
   arrData[sizeOfArry-1]=NULL;  }
```

```
Array A(20);

A.removeFromTop();
```

Lab 01 : Answers

Function to Remove an element at the end

```
void Array::removeFromBottom()
{
    arrData[sizeOfArry-1]=NULL;
}
```

```
Array A(20);

A.removeFromBottom();
```

Function to Inverse the order of the elements in the array

```
void Array::reverseArray()
{
    int temp;

    for(int i=0; i<sizeOfArry/2;i++)
    {
      temp = arrData[i];
      arrData[i] = arrData[sizeOfArry-i-1];
      arrData[sizeOfArry-i-1] = temp;
    }
}
```

```
Array A(20);

A.reverseArray();
```

Function to Return the sum of the elements in the array

```
int Array::returnSum()
{
   int Sum=0;
   for(int i =0;i<sizeOfArry;i++)
      Sum +=arrData[i];
}
```

```
Array A(20);

cout<<"Sum:  "<<A.returnSum()<<endl;
```

Function to Return an array that contains the odd numbers only

```
int* Array::checkOdd()
{
   bool isOdd =true;
    for(int i =0;i<sizeOfArry;i++)
      { if(arrData[i]%2==0){isOdd =false;
break;}}

      if(isOdd)return arrData;
      else return NULL;
}
```

```
Array A(20);
int *DD;

DD = A.checkOdd();
```

Lab 01 : Answers

Function to Display the content of the array in a file through fstream

```
void Array::writeToFile(char* filename)
{
    ofstream myfile;
    myfile.open (filename);
    for(int i =0;i<sizeOfArry;i++)
      {
         myfile << arrData[i] <<"\n";
      }
    myfile.close();
}
```

```
Array A(20);


A.writeToFile("Data.txt");
```

Overload the + operator so you can concatenate 2 arrays of size n and m into a new array of size n+m

```
int* Array::operator+(const Array& M) const
{

   int totalSize = this->sizeOfArry
   +M.sizeOfArry;

    int *result =  new int[totalSize];

     for(int i=0;i<totalSize;i++)
     {
         if(i<this->sizeOfArry)
            result[i]=this->arrData[i];
         else
            result[i]=M.arrData[i-this->
   sizeOfArry];
     }

     return result;

}
```

```
Array A(20);
Array B(10);
int *DD;


DD = A+B;

for(int i=0; i<(A.getSize()+B.getSize());
i++)
        cout<<DD[i]<<endl;
```