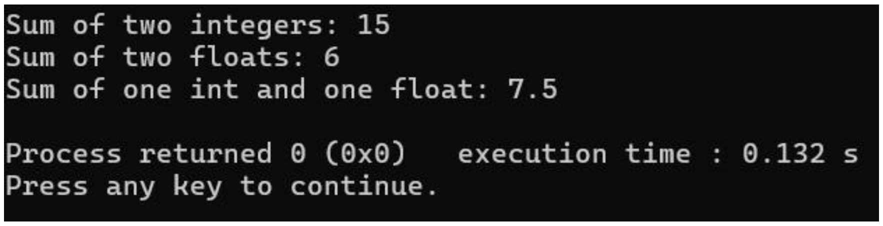
LAB QUESTIONS

Q1. Write a C++ program to overload a function add() to handle: Two integers, Two floats and One integer and one float.

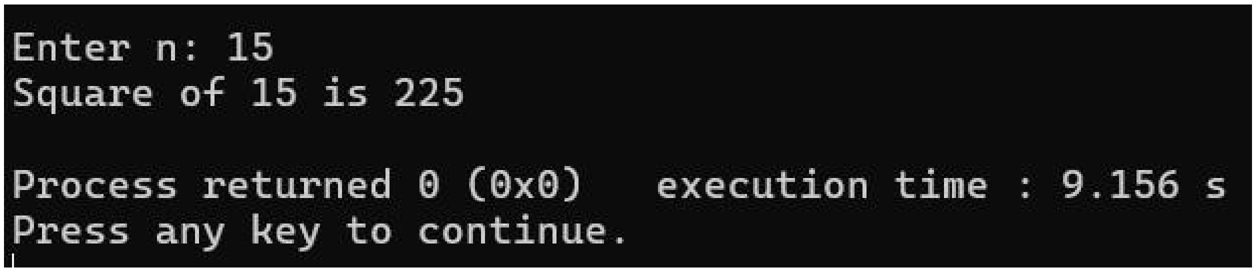
```
#include <iostream>

using namespace std;

int add(int a, int b) {

return a + b;

}

float add(float x, float y) {

return x + y;

}

float add(int a, float b) {

return a + b;

}

int main() {

int a = 5, b = 10;

float x = 3.5f, y = 2.5f;

cout << "Sum of two integers: " << add(a, b) << endl;

cout << "Sum of two floats: " << add(x, y) << endl;

cout << "Sum of one int and one float: " << add(a, y)

<< endl;

return 0;}
```



```
Sum of two integers: 15
Sum of two floats: 6
Sum of one int and one float: 7.5

Process returned 0 (0x0)   execution time : 0.132 s
Press any key to continue.
```

Q2. Write an inline function in C++ to calculate the square of a number and demonstrate it with at lease two function calls.

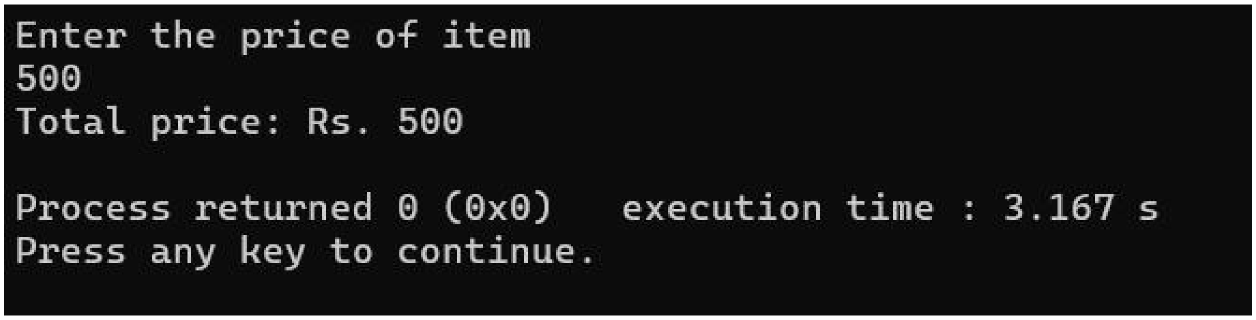
```
##include <iostream>

using namespace std;

inline int square(int n) {

   return n * n;

}

int main() {

   int n;

   cout << "Enter n: ";

   cin >> n;

   cout << "Square of " << n << " is " << square(n) << endl;

   return 0;

}
```



```
Enter n: 15
Square of 15 is 225

Process returned 0 (0x0)   execution time : 9.156 s
Press any key to continue.
```

Q3. Write a program using a function with default arguments for calculating total price. The function should take the item price and quantity, with quantity defaulting to 1.

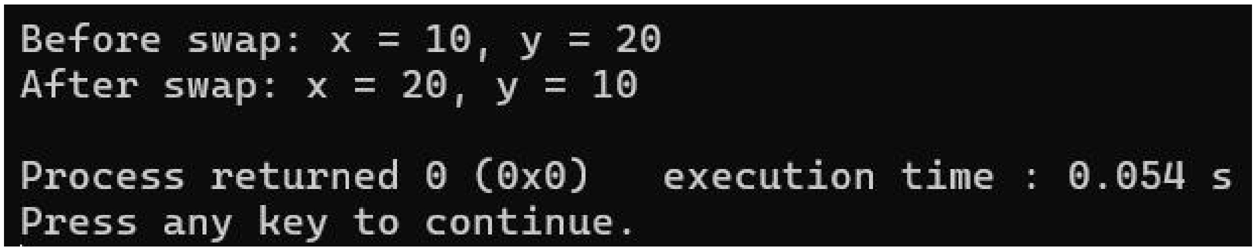
```
#include <iostream>
using namespace std;
float calculateTotal(float price, int quantity = 1) {
   return price * quantity;
}
int main() {
   float itemPrice;
   cout << "Enter the price of item" << endl;
   cin >> itemPrice;
   cout << "Total price: Rs. " << calculateTotal(itemPrice, 1) << endl;
   return 0;
}
```



```
Enter the price of item
500
Total price: Rs. 500

Process returned 0 (0x0)   execution time : 3.167 s
Press any key to continue.
```

Q4. Write a C++ program to swap two numbers using pass-by-reference.

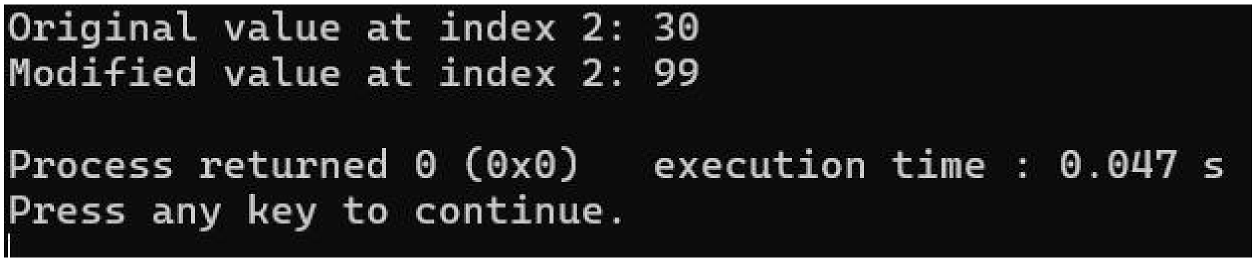
```
#include <iostream>
using namespace std;
void swapNumbers(int &a, int &b) {
int temp = a;
a = b;
b = temp;
}
int main() {
int x = 10, y = 20;
cout << "Before swap: x = " << x << ", y = " << y << endl;
swapNumbers(x, y);
cout << "After swap: x = " << x << ", y = " << y << endl;
return 0;
}
```



```
Before swap: x = 10, y = 20
After swap: x = 20, y = 10

Process returned 0 (0x0)   execution time : 0.054 s
Press any key to continue.
```

Q5. Create a function that returns a reference to an element in an array and modifies it.

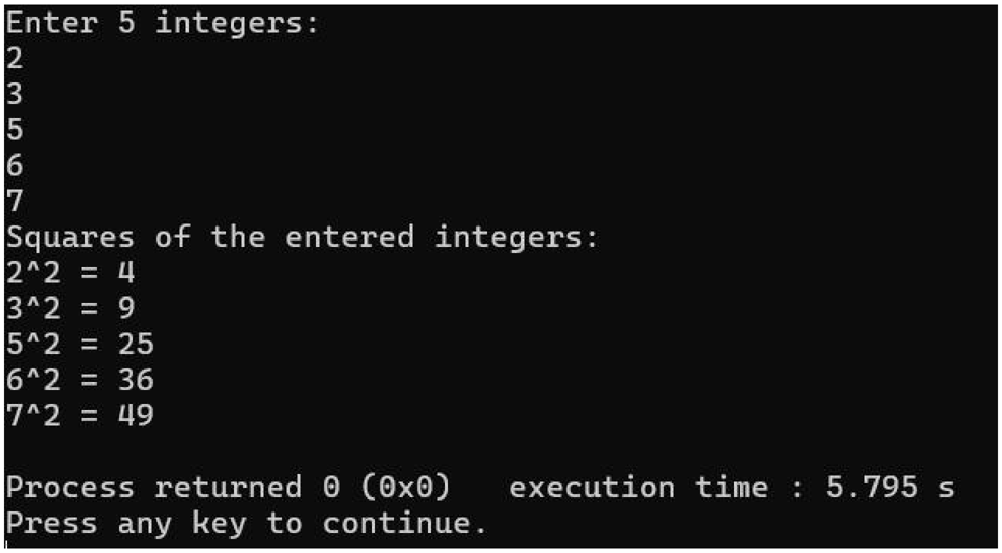
```
 #include <iostream>
using namespace std;
int& getElement(int arr[], int index) {
return arr[index]; // returning reference
}
int main() {
int numbers[5] = {10, 20, 30, 40, 50};
cout << "Original value at index 2: " << numbers[2] << endl;
getElement(numbers, 2) = 99;
cout << "Modified value at index 2: " << numbers[2] << endl;
return 0;
}
```



```
Original value at index 2: 30
Modified value at index 2: 99

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

Q6. Write a program to input 5 integers in an array and print their squares using a pointer.

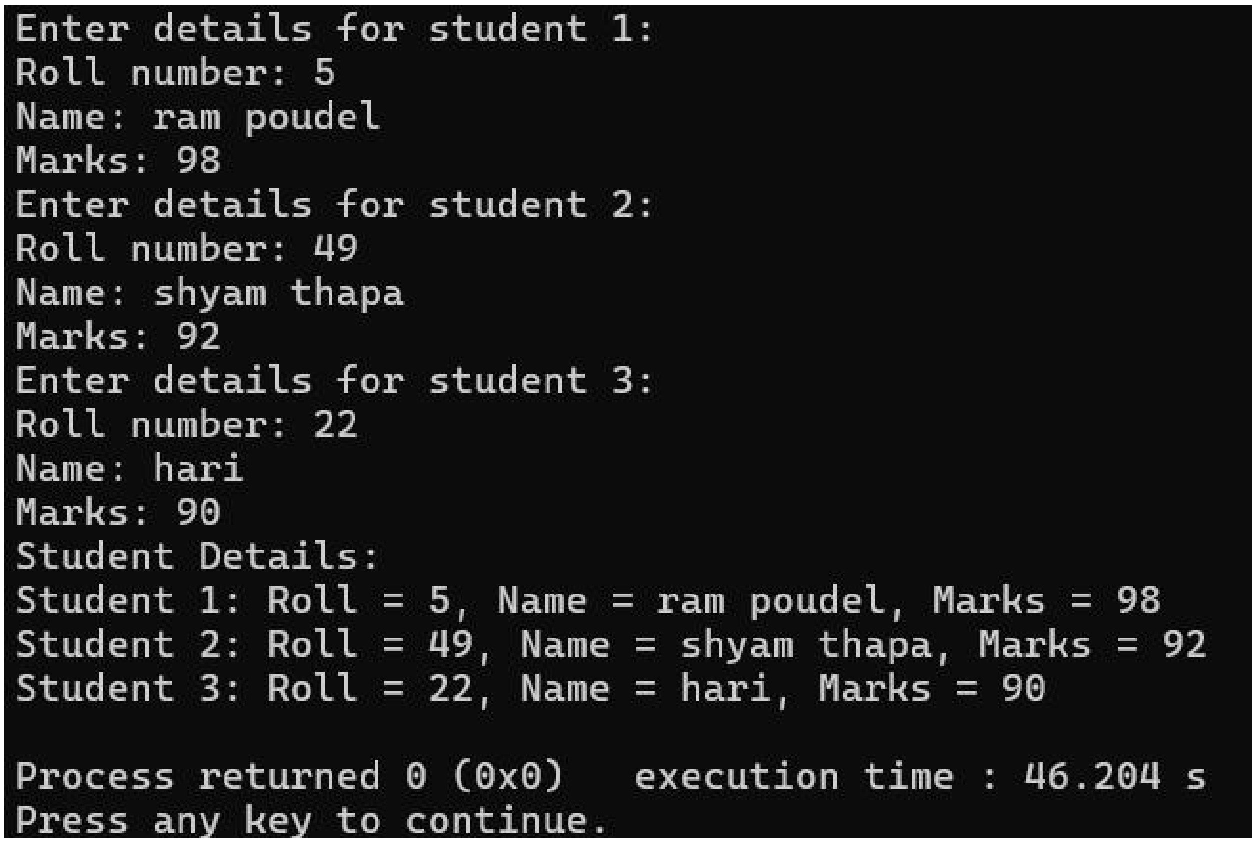
```
#include<iostream>
using namespace std;
int main() {
int arr[5];
int* ptr = arr;
// Input 5 integers
cout << "Enter 5 integers:”<<endl;
for (int i = 0; i < 5; i++) {
cin >> *(ptr + i);
}
cout << "Squares of the entered integers:"<<endl;
for (int i = 0; i < 5; i++) {
cout << *(ptr + i) << "^2 = " << (*(ptr + i)) * (*(ptr + i)) << endl;
}
return 0;}
```



```
Enter 5 integers:
2
3
5
6
7
Squares of the entered integers:
2^2 = 4
3^2 = 9
5^2 = 25
6^2 = 36
7^2 = 49

Process returned 0 (0x0)   execution time : 5.795 s
Press any key to continue.
```

Q7. Define a structure Student with data members roll, name, and marks. Input and display details of 3 students.

```
#include <iostream>

using namespace std;

struct Student {

int roll;

string name;

float marks;

};

int main() {

Student s[3];

for (int i = 0; i < 3; i++) {

cout << "Enter details for student " << i + 1 << ":"<<endl;

cout << "Roll number: ";

cin >> s[i].roll;

cin.ignore();

cout << "Name: ";

getline(cin, s[i].name);

cout << "Marks: ";

cin >> s[i].marks;

}

cout << "Student Details:"<<endl;

for (int i = 0; i < 3; i++) {

cout << "Student " << i + 1 << ": ";

cout << "Roll = " << s[i].roll << ", Name = " << s[i].name <<", Marks = " << s[i].marks << endl;

return 0;

}
```

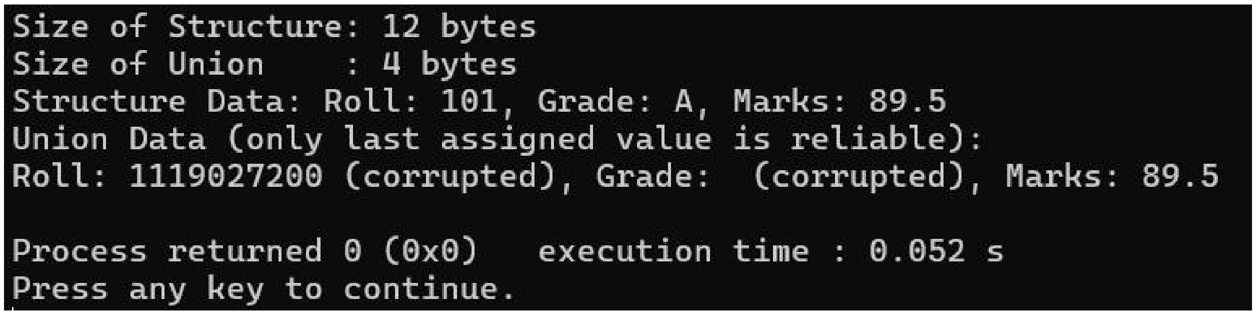


```
Enter details for student 1:
Roll number: 5
Name: ram poudel
Marks: 98
Enter details for student 2:
Roll number: 49
Name: shyam thapa
Marks: 92
Enter details for student 3:
Roll number: 22
Name: hari
Marks: 90
Student Details:
Student 1: Roll = 5, Name = ram poudel, Marks = 98
Student 2: Roll = 49, Name = shyam thapa, Marks = 92
Student 3: Roll = 22, Name = hari, Marks = 90

Process returned 0 (0x0)   execution time : 46.204 s
Press any key to continue.
```

Q8. Write a C++ program to demonstrate the difference between structure and union by declaring the same data members and showing memory usage.

```
#include <iostream>
using namespace std;
struct StudentStruct {
    int roll;
    char grade;
    float marks;
};
union StudentUnion {
    int roll;
    char grade;
    float marks;
};
int main() {
    StudentStruct s;
    StudentUnion u;
    cout << "Size of Structure: " << sizeof(s) << " bytes" << endl;
    cout << "Size of Union    : " << sizeof(u) << " bytes" << endl;
    s.roll = 101;
    s.grade = 'A';
    s.marks = 89.5;
    cout << "Structure Data: ";
    cout << "Roll: " << s.roll << ", Grade: " << s.grade << ", Marks: " << s.marks << endl;
    u.roll = 101;
    u.grade = 'A';
    u.marks = 89.5;
cout << "Union Data (only last assigned value is reliable):" << endl;
```

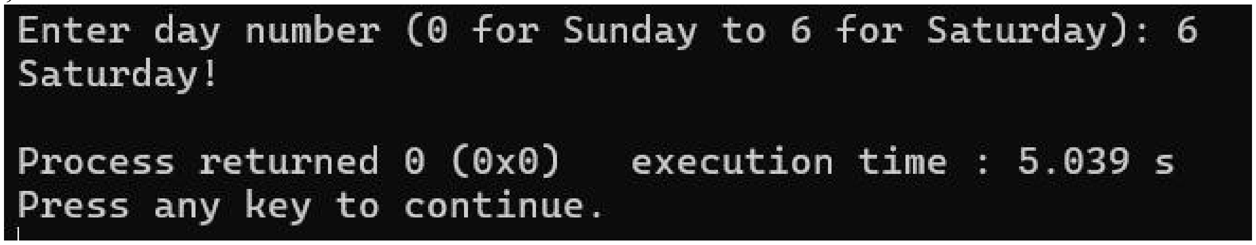
```
    cout << "Roll: " << u.roll << " (corrupted), Grade: " << u.grade << " (corrupted), Marks: " <<
u.marks << endl;

    return 0;

}
```



```
Size of Structure: 12 bytes
Size of Union    : 4 bytes
Structure Data: Roll: 101, Grade: A, Marks: 89.5
Union Data (only last assigned value is reliable):
Roll: 1119027200 (corrupted), Grade:  (corrupted), Marks: 89.5

Process returned 0 (0x0)   execution time : 0.052 s
Press any key to continue.
```

Q9. Create an enum for days of the week. Display a message depending on the selected day.

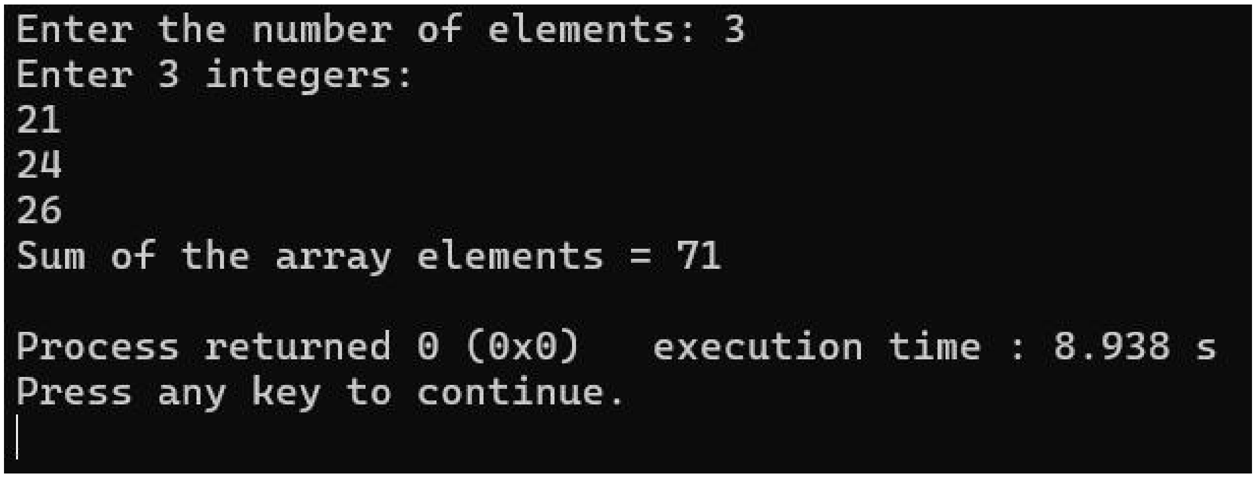
```
#include <iostream>
using namespace std;
enum Day { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday,Saturday };
int main() {
Day today;
int choice;
cout << "Enter day number (0 for Sunday to 6 for Saturday): ";
cin >> choice;
today = static_cast<Day>(choice);
switch (today) {
case Sunday:
cout << "Sunday." << endl;
break;
case Monday:
cout << "Monday!" << endl;
break;
case Tuesday:
cout << "Tuesday!" << endl;
break;
case Wednesday:
cout << "wednesday!" << endl;
break;
case Thursday:
cout << "thursday!" << endl;
break;
case Friday:
cout << "friday!" << endl;
break;
case Saturday:
cout << "Saturday!" << endl;
break;
default:
cout << "Invalid day number." << endl;
}
return 0;
}
```



```
Enter day number (0 for Sunday to 6 for Saturday): 6
Saturday!

Process returned 0 (0x0)   execution time : 5.039 s
Press any key to continue.
```

Q10. Write a C++ program to allocate memory for an array of integers using new, input values, calculate their sum, and free the memory using delete.

```
#include <iostream>
using namespace std;
int main() {
int n;
cout << "Enter the number of elements: ";
cin >> n;
int* arr = new int[n];
cout << "Enter " << n << " integers:" << endl;
for (int i = 0; i < n; i++) {
cin >> arr[i];
}int sum = 0;
for (int i = 0; i < n; i++) {
sum += arr[i];
}
cout << "Sum of the array elements = " << sum <<endl;
delete[] arr;
return 0;
}
```



```
Enter the number of elements: 3
Enter 3 integers:
21
24
26
Sum of the array elements = 71

Process returned 0 (0x0)   execution time : 8.938 s
Press any key to continue.
```

DISCUSSIONS

In this lab session, we learned the basics of object-oriented programming in C++ by working with classes, constructors, and destructors. We created classes to define data and functions, then used objects to access them. Constructors were used to initialize objects automatically, and we practiced using default, parameterized, and copy constructors. We also implemented destructors to understand how resources are cleaned up when objects go out of scope. This session gave us a clear understanding of classes, objects, constructors and destructors.

CONCLUSIONS

This lab made the concepts of classes, constructors, and destructors much clearer. We got hands-on experience with creating and initializing objects, and saw how destructors work when objects go out of scope.