Q.1:

#include <iostream>

using namespace std;

int main()

{

int size = 5;

int\* dynamicArray = new int[size];

for (int i = 0; i < size; i++)

{

dynamicArray[i] = i \* 2;

}

cout << "Array elements using pointer arithmetic:\n";

for (int i = 0; i < size; i++)

{

cout << \*(dynamicArray + i) << ' ';

}

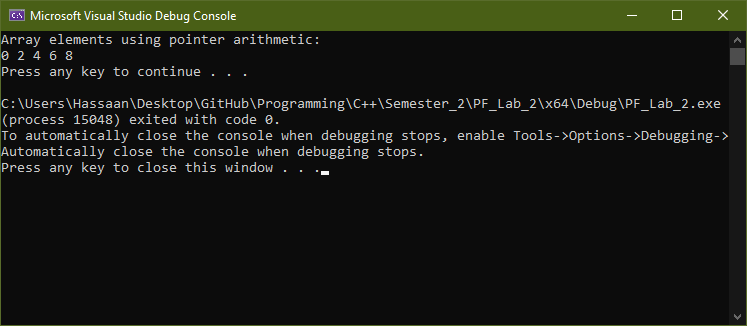
delete[] dynamicArray;

cout << endl;

system("pause");

return 0;

}



Q.2:

#include <iostream>

using namespace std;

int main()

{

int\* dynamicInteger = new int;

cout << "Enter integer value: ";

cin >> \*dynamicInteger;

cout << "Value using dereference operator: " << \*dynamicInteger << endl;

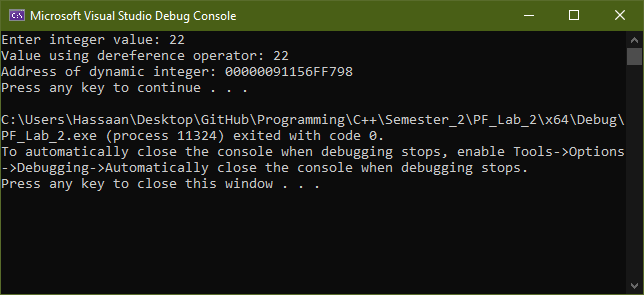
cout << "Address of dynamic integer: " << &dynamicInteger << endl;

delete dynamicInteger;

system("pause");

return 0;

}



Q.3:

#include <iostream>

using namespace std;

int main()

{

int size;

cout << "Enter size of array: ";

cin >> size;

int\* dynamicArray = new int[size];

cout << "Enter " << size << " integers for the array:\n";

for (int i = 0; i < size; i++)

{

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

cin >> dynamicArray[i];

\*(dynamicArray + i) += 5;

}

cout << "Modified array after incrementing elements by 5:\n";

for (int i = 0; i < size; i++)

{

cout << \*(dynamicArray + i) << ' ';

}

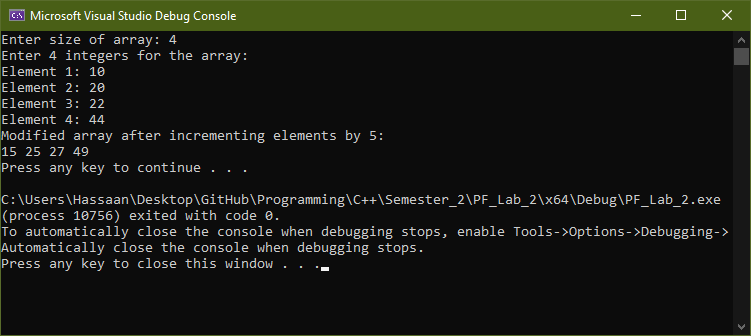
delete[] dynamicArray;

cout << endl;

system("pause");

return 0;

}



Q.4:

#include <iostream>

using namespace std;

void reverseArray(int\* arr, int size)

{

for (int i = 0; i < size / 2; i++)

{

int temp = \*(arr + i);

\*(arr + i) = \*(arr + size - 1 - i);

\*(arr + size - 1 - i) = temp;

}

}

int main()

{

int size;

cout << "Enter size of array: ";

cin >> size;

int\* dynamicArray = new int[size];

cout << "Enter " << size << " integers for the array:\n";

for (int i = 0; i < size; i++)

{

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

cin >> dynamicArray[i];

}

reverseArray(dynamicArray, size);

cout << "The reversed array is:\n";

for (int i = 0; i < size; i++)

{

cout << \*(dynamicArray + i) << ' ';

}

delete[] dynamicArray;

cout << endl;

system("pause");

return 0;

}

