COLLEGE OF INFORMATICS AND VIRTUAL EDUCATION(CIVE)

Department of computer science and engineering

GROUP ASSIGNMENT:

CLASS: BSc.SE

COURSE CODE: CP 213

NAME	REGISTRATION NUMBER
BONIFACE ELIAH GWAKILA	T22-03-04314
ASHA SELEMAN KIZITO	T22-03-11747
ELIAS JOSEPH MANJALE	T22-03-00709
RICHARD PAULO NHONYA	T22-03-13558
DERICK GABRIEL MHIDZE	T22-03-04321

Qn 1:Pointer is a variable that stores the memory address of the another variable.

Meaning that it provides the location in the memory for specific value to be stored, it is declared using symbols like *, & with different meaning each with its meaning such as *x it assigns the value stored at the memory address. &x just initializes a pointer to the address of integer variable x.

```cpp
pointer.cpp
1    #include<iostream>
2    using namespace std;
3    int main(){
4        //variable declaration
5        int group1 =100;
6        cout<<"Value of th variable group1 is: "<<group1<<endl;
7        //assign group1 variable address to the pointer.
8        int *ptr=&group1;
9        cout<<"The address of the variable group1 is: "<<ptr<<endl;
10       //pointer variable change the value of variable referencing to it
11       *ptr=200;
12       //now the value of the variable group 1 is 200
13       cout<<"Value of th variable group1 is: "<<group1<<endl;
14       //but the address still the same
15       cout<<"The address of the variable group1 still: "<<ptr<<endl;
16
17
18   return 0;
19   }
```

```
C:\Users\MICROSPACE\Desktop\assgment\pointer.exe
Value of th variable group1 is: 100
The address of the variable group1 is: 0x6ffe04
Value of th variable group1 is: 200
The address of the variable group1 still: 0x6ffe04

--------------------------------
Process exited after 0.4517 seconds with return value 0
Press any key to continue . . .
```

Qn 2: Static memory allocation occur when memory is allocated at compile time the size of the memory is determined before the program runs. The memory is typically allocated for variable declared at the beginning of the program or in a specific function

staticMemoryAllocation.cpp

```cpp
1    #include<iostream>
2    using namespace std;
3    int main(){
4        //static memory allocation for variable
5        int y=10;
6        //static memory allocation for array variables
7        int staticAlloc[5]={4,6,8,9,5};
8        int sum=0;
9        for(int i=0;i<5;i++){
10          sum +=staticAlloc[i];
11       }
12       cout<<"sum is: "<<sum;
13
14
15   return 0;
16   }
```

C:\Users\MICROSPACE\Desktop\assgment\staticMemoryAllocation.exe

```
sum is: 32
-----------------------------------
Process exited after 0.475 seconds with return value 0
Press any key to continue . . .
```

: Dynamic memory allocation it involves allocating memory at run time and the size of the allocated memory can be determined during program execution, and it's done using operator like 'new' and 'delete'. Where 'new' allow to add even during run time no matter the size you allocate before. 'delete' just help programmers to free the allocated memory .

dynamicAlloc.cpp

```cpp
1    #include <iostream>
2    using namespace std;
3    int main() {
4        // Dynamic array with size determined at runtime
5        int size;
6        cout << "Enter the size of the dynamic array: ";
7        cin >> size;
8
9        // Dynamic memory allocation
10       int* dynamicArray = new int[size];
11
12       // Initializing elements
13       for (int i = 0; i < size; ++i) {
14           dynamicArray[i] = i + 1;
15       }
16
17       // Accessing elements
18       for (int i = 0; i < size; ++i) {
19           cout << dynamicArray[i] << " ";
20       }
21
22       //free the allocated memory
23       delete[] dynamicArray;
24
25       return 0;
26   }
```

```
Enter the size of the dynamic array: 6
1 2 3 4 5 6
-----------------------------------
Process exited after 3.151 seconds with return value 0
Press any key to continue . . .
```

Qn 3: Static array just arrays which have fixed size which are determined during compile-time such they are declared with specific size, and the memory is allocated on the stack. In such a sense that are suitable when the size is constant and known beforehand

staticMemoryAllocation.cpp

```
1    #include<iostream>
2    using namespace std;
3    int main(){
4        //static memory allocation for variable
5        int y=10;
6        //static memory allocation for array variables
7        int staticAlloc[5]={4,6,8,9,5};
8        int sum=0;
9        for(int i=0;i<5;i++){
10         sum +=staticAlloc[i];
11       }
12       cout<<"sum is: "<<sum;
13
14
15   return 0;
16   }
```

C:\Users\MICROSPACE\Desktop\assgment\staticMemoryAllocation.exe

```
sum is: 32
--------------------------------
Process exited after 0.475 seconds with return value 0
Press any key to continue . . .
```

: Dynamic arrays are those having size that can be determined at runtime in such a way that they are created using pointers and memory allocation operators like 'new' and 'delete'. And are suitable when the size is not known at compile-time or when the size may change during program execution

```cpp
dynamicAlloc.cpp
1    #include <iostream>
2    using namespace std;
3    int main() {
4        // Dynamic array with size determined at runtime
5        int size;
6        cout << "Enter the size of the dynamic array: ";
7        cin >> size;
8
9        // Dynamic memory allocation
10       int* dynamicArray = new int[size];
11
12       // Initializing elements
13       for (int i = 0; i < size; ++i) {
14           dynamicArray[i] = i + 1;
15       }
16
17       // Accessing elements
18       for (int i = 0; i < size; ++i) {
19           cout << dynamicArray[i] << " ";
20       }
21
22       //free the allocated memory
23       delete[] dynamicArray;
24
25       return 0;
26   }
```

```
Enter the size of the dynamic array: 6
1 2 3 4 5 6
------------------------------
Process exited after 3.151 seconds with return value 0
Press any key to continue . . .
```

Qn 4: Iteration these involves repeated execution of set of statement or block of code. It is used for tasks that needs to be performed at a specific number of times or until a certain condition is met. We achieve it by a way o called looping which involves FOR, WHILE, DO WHILE. It is suitable for many tasks.

```cpp
iteration.cpp
1    #include <iostream>
2    using namespace std;
3    int main() {
4        int y=0;
5        while(y<5){
6            cout<<y<<endl;
7            ++y;
8        }
9
10       return 0;
11   }
```

```
0
1
2
3
4

---------------------------------
Process exited after 0.3105 seconds with return value 0
Press any key to continue . . .
```

: Recursion as a programming concept where a function calls it self in order to solve a smaller instance of the same problem. It involves breaking down of the problem into smaller subproblems and solve them until reaching a base case, it requires careful design to ensure it converges to the base case and doesn't result in infinite recursion.

recursion.cpp

```cpp
1    #include <iostream>
2    using namespace std;
3    int factorialNumber(int number){
4        //base condition to stop function to run
5        if(number==0){
6            return 1;
7        }
8        return number * factorialNumber(number-1);
9    }
10   int main() {
11       int z=6;
12       int y=factorialNumber(z);
13       cout<<"Factorial of "<<z<<" is "<<y;
14
15       return 0;
16   }
17
```

C:\Users\MICROSPACE\Desktop\assgment\recursion.exe

```
Factorial of 6 is 720
---------------------------------
Process exited after 0.451 seconds with return value 0
Press any key to continue . . .
```

Qn 5: A sentinel loop refers to the loop that continues to execute until a specific condition is met, typically involves the use of special value known as sentile value that signal the end of

the loop. And are commonly used when the exact number of iterations is not known beforehand and depends on user input or external factors and just provide flexible way to handle varying amounts of input data while ensuring a clear termination condition

```cpp
cp2133.cpp
1    #include<iostream>
2    using namespace std;
3    int main(){
4        const int sentinel=2;
5        int num;
6        int sum=0;
7        while(true){
8            cin>>num;
9            if (num==sentinel){
10               break;
11           }
12           sum +=num;
13
14       }
15       cout<<"sum of entered number is"<<sum<<endl;
16
17   }
```

C:\Users\MICROSPACE\Documents\cp2133.exe

```
6
5
6
8
9
0
2
sum of entered number is34

------------------------------
Process exited after 15.91 seconds with return value 0
Press any key to continue . . .
```

The program runs in such a way that when the condition met for value 2 it terminate but already provide the answer for the sum.