Assignment No:3

HAFIZ NAQASH

ID 793-2022

Sub PF

1 What are Functions and types?

2 Write down the program of functions?

3 What is stack rolling and unrolling?

4 What is string and string operations?

5 Write down the program of string?

6 What is static and dynamic memory allocation?

7 Write down the program static and dynamic allocation?

8 What are file I/O operations?

9 Write down the program logical operator?

10 Write down the program nested loop?

1 What are Functions and types?

In programming, a function is a block of organized, reusable code that is used to perform a single, related action. Functions typically take one or more input values (called parameters or arguments) and return an output value. Functions are often used to encapsulate a specific piece of logic or computation, making it easier to understand, test, and maintain a program.

There are many different types of functions, including:

Built-in functions: These are functions that are built into the programming language itself and are available to use without the need to define them. Examples include print() in Python, or len() in python.

User-defined functions: These are functions that are defined by the user in their code. They are created using the 'def' keyword in Python.

Anonymous functions: These are functions that are defined without a name, also called lambda functions. They can be useful when you need to pass a small function as an argument to another function.

Recursive functions: These are functions that call themselves. These are very useful when solving problems that can be broken down into smaller subproblems.

2 Write down the program of functions?

Here is an example of a simple program in Python that demonstrates the use of a user-defined function:

```
def greet(name):
    """This function greets the person passed in as a parameter"""
    print(f"Hello, {name}!")

greet("John")
greet("Mary")
```

This program defines a function called greet() that takes in a single parameter name. Inside the function, the code uses the print() function to output a greeting to the screen using the passed-in name.

The program then calls the greet() function twice, passing in the strings "John" and "Mary" as arguments. The output of the program will be:

```
Hello, John!
Hello, Mary!
```

Another example could be

```
def add(a, b):
    """This function returns the sum of two numbers"""
    return a + b
```

```
result = add(3, 4)

print(result)
```

This program defines a function called add() that takes in two parameters a, and b. Inside the function, we return the sum of a and b.

The program then calls the add() function with the arguments 3 and 4 and store the returned value in the variable result. The output of the program will be 7

3 What is stack rolling and unrolling?

Stack rolling refers to a technique in which the memory of a computer's stack, which stores information about function calls and their execution, is used in a circular or cyclical manner. This can be done to save memory or to implement certain types of algorithms.

Unrolling a stack refers to the process of removing or "popping" elements from the top of the stack in a specific order. Unrolling is often used to reverse the order of elements on the stack or to access elements that are deeper in the stack without having to pop all the elements above them.

4 What is string and string operations?

A string is a sequence of characters in a program. In many programming languages, strings are enclosed in quotation marks (either single or double) to distinguish them from other types of data. Strings are commonly used to represent text, such as words, sentences, and paragraphs.

There are several common operations that can be performed on strings, including:

Concatenation, which is the process of joining two or more strings together to form a new string

Substring extraction, which is the process of extracting a portion of a string

Searching, which is the process of finding a specific character or substring within a string

Replacing, which is the process of replacing a specific character or substring within a string with another character or substring

Sorting, which is the process of arranging the characters in a string in a specific order

Formatting, which is the process of aligning, spacing and/or styling for output presentation

tokenizing, which is the process of breaking a string into smaller pieces or tokens.

Most modern languages include a built-in string data type and associated libraries that provide functions to perform these and other operations on strings.

5 Write down the program of string?

There are many different operations that can be performed on strings in a computer program, depending on the programming language and libraries available. Some common string operations include:

Concatenation: combining two or more strings into a single string

Substring: extracting a portion of a string

Trimming: removing whitespace or other characters from the beginning or end of a string

Replacing: replacing all occurrences of one substring with another

Splitting: breaking a string into an array of substrings

Searching: finding the index of a substring within a string

Formatting: manipulating the capitalization or layout of a string

Encoding/Decoding: converting string to bytes, bytes to string

Regular Expressions: pattern matching with strings

Here is an example of some string operations in the Python programming language:

```
string1 = "Hello, "
```

```
string2 = "world!"

# Concatenation
result = string1 + string2
print(result)  # "Hello, world!"

# Substring
result = string1[2:5]
print(result) # "llo"

# Trimming
string3 = "   trim me   "
result = string3.strip()
print(result) # "trim me"

# Replacing
result = string1.replace("l","X")
print(result) # "HeXlo, "

# Splitting
string4 = "apples,oranges,bananas"
result = string4.split(",")
print(result) # ["apples", "oranges", "bananas"]

# Searching
```

```
result = string1.find("l")

print(result) # 2


# Formatting

result = string2.upper()

print(result) # "WORLD!"


#Encoding

string5 = "Hello World"

result = string5.encode()

print(result) # b'Hello World'


#Decoding

result = result.decode()

print(result) # "Hello World"
```

It's important to note that this is just a small sample of the different string operations that are available in many programming languages, and the specific syntax and functionality may vary depending on the language you're using.

6 What is static and dynamic memory allocation?

Static memory allocation refers to the allocation of memory at compile-time, before the program is executed. This means that the amount of memory allocated is fixed and determined when the program is written. For example, in the C programming language, variables declared as "static" or global variables are allocated memory in the static memory area.

Dynamic memory allocation refers to the allocation of memory at runtime, during the execution of the program. This means that the amount of memory allocated can be determined and changed while the

program is running. For example, in the C programming language, the malloc() and calloc() functions are used to dynamically allocate memory from the heap.

In general, static memory allocation is simpler and less error-prone than dynamic memory allocation, but it can be less flexible, since the amount of memory allocated is fixed at compile-time. Dynamic memory allocation allows for more flexibility, but it can be more error-prone since it is the programmer's responsibility to correctly manage the memory that has been allocated.

7 Write down the program static and dynamic allocation?

Static allocation is the allocation of memory at compile-time, before the program is executed. This means that the amount of memory allocated for a variable or data structure is fixed and determined at the time the program is written.

An example of static allocation in C is:

```
int myArray[100];
```

In this example, 100 integers worth of memory are reserved for the variable myArray at compile time, regardless of the program's runtime behavior.

Dynamic allocation, on the other hand, is the allocation of memory at runtime, during the execution of the program. This means that the amount of memory allocated for a variable or data structure can be determined and changed while the program is running.

An example of dynamic allocation in C is:

```
int *myArray = malloc(100 * sizeof(int));
```

In this example, the malloc() function is used to dynamically allocate 100 integers worth of memory and the return address is assigned to the pointer myArray.

You can use this allocated memory until you use free() function to deallocate.

8 What are file I/O operations?

File input/output (I/O) operations are the means by which a program can read and write data to files on a storage device, such as a hard drive. This can include reading data from an existing file, writing data to a new or existing file, and modifying or deleting data stored in a file. The specific syntax and functionality for file I/O operations varies depending on the programming language being used, but generally involves a set of built-in functions or libraries that a programmer can use to open, read, write, and close files.

9 write down the program on logical operator?

In computer programming, logical operators are used to perform logical operations on boolean variables (i.e. variables that can only have the values true or false). The most common logical operators are:

AND (&& in C-like languages, and in Python), which returns true if both its operands are true, and false otherwise.

OR (|| in C-like languages, or in Python), which returns true if at least one of its operands is true, and false otherwise.

NOT (! in C-like languages, not in Python), which negates its operand, i.e. it returns true if its operand is false, and false if its operand is true.

For example:

a = true

b = false

c = (a && b) // c is false

d = (a || b) // d is true

e = !a // e is false

In addition to that we have bitwise operator such as

-& : Bitwise AND

-| : Bitwise OR

-^ : Bitwise XOR

-~ : Bitwise NOT

-<< : Bitwise Left Shift

->> : Bitwise Right Shift

Example:

```
x = 60;   // 60 = 0011 1100

y = 13;   // 13 = 0000 1101

z = x & y;       // 12 = 0000 1100
```

10 Write down the program nested loop?

```
for i in range(1, 3):
    for j in range(1, 4):
        print(f"i = {i}, j = {j}")
```

This program will output the following:

```
i = 1, j = 1

i = 1, j = 2

i = 1, j = 3

i = 2, j = 1

i = 2, j = 2

i = 2, j = 3
```

The outer for loop iterates over the range of 1 to 3 (not inclusive of 3) and the inner for loop iterates over the range of 1 to 4 (not inclusive of 4). The print statement will be called for each iteration of the inner loop, resulting in the above output.