Simulation of Co-Routines

We all are familiar with function which is also known as a subroutine, procedure etc, in all programming languages.

A function is a sequence of instructions packed as a unit to perform a certain task. When the logic of a complex function is divided into several self-contained steps that are themselves functions, then these functions are called subroutines. For example in C main() is routine but it is special routine which is also called the entry point. This routine calls other functions which we call as sub routine.

Some programming languages (like python) supports a feature called Co-routines. What are Co-routines

Co routines are sub routines in general. They are used for cooperative multitasking where a process voluntarily yield (give away) control periodically or when idle in order to enable multiple applications to be run simultaneously.

The difference between coroutine and subroutine is :

Unlike subroutines, coroutines will have many points where the execution can be suspended and resumed at various pre-defined points there by giving way to processor to execute another co-routine or to some other program.

Coroutine can suspend its execution and transfer control to other coroutine and can resume again execution from the point it left off.

Unlike subroutines, there is no main function to call coroutines in particular order and coordinate the results. Coroutines are cooperative that means they link together to form a pipeline. One coroutine may consume input data and send it to other which process it. Finally there may be a coroutine to display result.

Co routines are real power concept especially when the programmer wants complete control over every point in scheduling rather than giving the control to a centralized scheduler which is based on an algorithm. This makes it a very power concept to design real time applications which run on cloud computers. This is because the control is with a central point than with a scheduler, then it is only a matter of adding the timing constraints. (We will discuss about cloud computers and its components later).

Here is an example on co-routine in Python

```
def print_name(prefix):
    print("Searching prefix:{}".format(prefix))
    try :
        while True:
                name = (yield)
                if prefix in name:
                    print(name)
    except GeneratorExit:
            print("Closing coroutine!!")

def main():
   handle = print_name("Dear")   # Call the co Routine first
   handle.__next__()            # Move to next
   handle.send("Mahesh")        # Send a message to co routine
   handle.send("Dear Mahesh")   # Send a message to co routine
   handle.send("Dear Anand")    # Send a message to co routine
   handle.send("Dr Anand")      # Send a message to co routine
   handle.close()               # Close the handle

if __name__ == '__main__' :
  main()
```

Reference : https://www.geeksforgeeks.org/

C as programming Language does not support co routines, hence it is not possible to make use of this concept in C directly. Since we use C programming in many of embedded system it will be nice if this feature is made available in C, but the C function calling stack does not allows this, it was not designed taking this kind of features in mind.

However, with the concept of process and threads it is possible to simulate this effect in C or C++ using the concepts we discussed in class. Your objective is simulate this using C on your laptop. Though this will not be an exact replacement for the concept of Co-Routines what we see in Python, implementation of a program to simulate co-routines will give you lot of exposure into the concepts behind this.

Note: The co-routine concept in Python is much bigger topic than we discussed above. What we talked above is few concepts that can be used for our exercise.

Requirements

1. It should be possible to simulate the start and stopping of a co-routine as shown in the above program.

2. We should be able to Send data to co-routine with something like handle.send() API (You need to implement this)

3.We should be able to pause and resume the execution of co-routines from a central routine.

4. We should be able to modify and existing routine written in C to a Co-routine pattern

5. We should be able to run and manage multiple co-routines in a single program.

6. Using this concept implement a snake and ladder game with four players. Where each player gets a chance to throw the dice get a random number between 0 and 7 and advance their position to next column the game continues till one of them reaches the max col 100. Have the starting position of snakes and ladders. This again will be simulation of the game with automatic players, each player will be co routine. Once starts the players will automatically play and game will get over within a specific time. Repeat this game again and again and find out the time taken for each round and tabulate it. Though snake and ladder is a very simple game this can be the first step towards modelling games with timing constraints.