EXPERIMENT NO. - 10

AIM: Write a function which takes a tuple as a parameter and returns a new tuple as the output, where every other element of the input tuple is copied, starting from the first one.

THEORY:

A tuple is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing −

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

```
#!/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
print "tup1[0]: ", tup1[0];
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result −

```
tup1[0]:  physics
tup2[1:5]:  [2, 3, 4, 5]
```

PROGRAM A)

```
def oddTuples(aTup): #Function with tuple as an argument
 rTup = () #Initially the output tuple rTup is empty
 index = 0
 while index < len(aTup):
 rTup += (aTup[index],)
 index += 2 #index increased by 2
 return rTup
t=(1, 3, 2, 4, 6, 5)
print(oddTuples(t))
```

Output:

(1, 2, 6)

Explanation In the above program, initially a tuple ‘t’ is created. This tuple ‘t’ is passed as a parameter to a function. The while loop iterates till the length of the tuple. In each iteration, the number stored at an odd index is accessed and stored into the output tuple ‘rTup’.

PROGRAM B)

```
#function definition
def odd_indices(tup):
   #create temporary tuples
   Odd_Tup = ()
   #iterate tuple
   for ind in range(len(tup)):
      #check if the element from the tuples gives the remainder
      if ind % 2 != 0:
         #if the number returns the remainder then that is an odd index
         #store that odd indexed value in a new tuple
         Odd_Tup += (tup[ind],)
   #return tuple
   return Odd_Tup

myTup = (1, 2, 3, 4, 5, 6)
print(odd_indices(myTup))
```

Explanation:

- The function takes one argument ‘tup’.
- Create another temporary tuple for keeping track of odd indexed values from the ‘tup’.
- Iterate tup using the loop, and the if statement check whether the index is odd or not.
- If the index gives a remainder then that is odd and store that index value in Odd_tup tuples
- Return the Odd_tup to the called function.

EXERCISE:

1. Write a function which takes a tuple as a parameter and returns a new tuple as the output, where every other element of the input tuple is copied, starting from the first one.

T = (‘Hello’,’Are’,’You’,’Loving’,’Python?’)
Output_Tuple = (‘Hello’, ‘You’, ‘Python?’)

Conclusion: Hence, we have successfully studied about program for writing a function which takes a tuple as a parameter and returns a new tuple as the output, where every other element of the input tuple is copied, starting from the first one.