**Computer Science Principles**

**Definition of terms**

1. Programming

The process of developing and implementing various sets of instructions (algorithms) to enable a computer to do a certain task

1. Algorithm

A well-defined procedure that allows a computer to solve a problem: for example a recipe

**Activity: Divide students into groups of 3 or 4. Ask them to write an algorithm to perform a task, eg. walk to the door.**

1. Variables

Variables are used to hold the information that varies, for example, if you want to enter various fields in a particular file, such as, student name, student ID, grade, etc. These fields hold the information that is different from one student to another.

Example

x = 2

y = 5

x = x + y

print (x)

1. Conditionals

Many a time you want to compare the information with regard to the fields entered. Such as, age, grade, gender etc. In order to make comparisons clear and understandable, computer programming involves some conditionals. For example, use of specific symbols, such > greater than, < lesser than, >= greater than equals to, <=less than equals to etc. Note that these symbols or conditionals are used with control flows using if then, if then else statements etc.

Example

num = input ("Please enter a number between 1 and 10: ")

num = int(num)

while num < 1 or num > 10:

num = input ("Invalid entry. Please enter a number between 1 and 10: ")

num = int(num)

if num <= 5:

print ("You have chosen a small number!")

else:

print ("You have chosen a large number!")

**Activity: Ask students to repeat the algorithm task using conditionals.**

Nested conditionals

There may be a situation when you want to check for another condition after a condition resolves to true. In such a situation, you can use the nested **if** construct. In a nested **if** construct, you can have an **if...elif...else** construct inside another **if...elif...else** construct.

1. Loops

Looping is one of the most important features of an effective computer programming by which a code causes the computer to do things repeatedly. Loop is a valuable tool, as with the help of loop, you can get the desired tasks done repeatedly or as many times as you want. For example, if you want to display serial numbers in a corresponding column from 1 to 100, this activity requires defining a loop in a computer program to do so.

Example

count = 1

while count <= 10:

print (count)

count = count + 1

1. Arrays

An array is a data structure that stores values of same data type. For example mailboxes on the same street.

Example

from array import \*

my\_array = array('i', [1,2,3,4,5])

*#Print array*

for i in my\_array:

print(i)

*#Add number to array*

my\_array.append(6)

for i in my\_array:

print(i)

*#Print index value in array*

print (my\_array[2])

**Explain the code below identifying functions of various components of the code**

# This program "gets smart" playing the game called nimb

Comments (use #)

# There are ten objects, the player and the computer can

# take one or two objects each turn. The one who takes the

# last object loses.

2 arrays

values = [0,0,0,0,0,0,0,0,0]

memory = [0,0,0,0,0,0,0,0,0]

#dhk int i, choices, selection, remaining, playagain, goodorbad;

userwins = 0

Variables (counter)

computerwins = 0

# Each repetition plays one game

Boolean (True or False)

playagain = True

while playagain:

remaining = 10

print ("Start Game. There are ", remaining, " objects")

print (" ")

Run as long as playagain = True

Initialize start of new game

choices = 0

# Loop, each repetition gives one move

while (remaining > 0):

selection = eval (input("How many objects do you take (1 or 2)? "))

if (selection < 1):

selection = 1

print("\b Illegal value, assuming 1")

Game loop while remaining > 0

elif (selection > 2):

selection = 2

print("\b Illegal value, assuming 2")

remaining = remaining - selection

if (remaining > 2): # did human leave more than two objects?

if (values[remaining-1] >= values[remaining-2]):

remaining = remaining - 1

else:

remaining = remaining - 2

print( "Computer leaves ", remaining)

choices = choices + 1

memory[choices] = remaining

elif (remaining == 2): # did human leave two objects?

remaining = 1

print("Computer leaves ", remaining)

choices = choices + 1

memory[choices] = remaining

elif (remaining == 1): # did human leave one object?

remaining = 0

print("Computer Loses")

goodorbad = -1

userwins = userwins + 1

else: # human left no objects

print( "Computer Wins")

goodorbad = 1

computerwins = computerwins + 1

# Increment or decrement the values of the positions the computer left

Storing choices in values and memory

for i in range (choices+1):

values[memory[i]] = values[memory[i]] + goodorbad

playagain = (input("Hit [Enter] to play again, any other key to exit: ") == "")

print("You won ", userwins, " times.")

print("I won ", computerwins, " times.")

print()

print("Thanks for playing, and teaching me how to win!")

**Activity:**

**Ask students to develop a sequence that always beats the computer.**

**Ask to students to identify a way to crash the game, why does it crash, and how to fix the crash.**

if (selection < 1):

selection = 1

If user enters a number greater than 1 and less than 2 (eg 1.3) the program crashes.

To fix change selection < 1 to selection < 2

print("\b Illegal value, assuming 1")

elif (selection > 2):

selection = 2

print("\b Illegal value, assuming 2")

remaining = remaining - selection