**25/03/2023 ASSESSMENT 01\_GOKULNATH\_K**

**Exercise#1: Find the maximum element in an array**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **23** | **65** | **12** | **87** | **12** | **343** | **098** | **12** | **76** | **312** |

**Pre-condition**

Define two variables:

A variable i that acts as a loop counter, and a variable max to store the maximum element of all integers.

Before starting the loop from arr[0] to arr[n-1], we need to initialize max=arr[0] and starts the loop from i=1

This precondition is true, enters the first iteration of loop

**Post-Condition**

After the loop termination, the max value must store the maximum value of all values from arr[0] to arr[n-1]

**Loop variant**

The loop must terminate after finding maximum value of all integers from arr[0] to arr[n-1]

In other words, loop should not terminate until compared arr[n-1] to max, i.e. i<=n-1 or i<n

**Loop invariant**

Let's assume invariant is true after (i-1)th iteration

i.e., max stores the maximum value of all integers from arr[0] to arr[n-1]

we need to design instruction so that the variant must be trye after ith iterations of the loop

i.e. max must be equal to the max of all integers from arr[0] to arr[i]

**Steps for the ith iteration**

compare max value with arr[i]. if(max<arr[i]), it means we found the greater value than max of all values from arr[0] to arr[n-1].

In such situation, update max value with arr[i] i.e. max=arr[i].

Otherwise, ignore it and value stored in max is still maximum from arr[n] to arr[i]

we should increase i by 1 at each iteration to update maximum from arr[0] to arr[n-1] in max

**Pseudocode**

int findMax(int arr[],int n)

{

int max = arr[0]

for(int i = 1; i < n; i++)

{

if(arr[i] > max)

max = arr[i]

}

return max

}

**Exercise#2: Convert roman numbers to integers**

**Pre-condition**

Define variables:

A variable ‘I’ that acts as a loop counter, and a variable total store the sum of integer values of char r.

A variable ‘s’ in the arguments to pass the input and a variables s1 and s2 to store the integer value at i and i+1th position.

Before starting the loop from i=0 to i<s.length(), we need to initialize total=0 and starts the loop.

This precondition is true, enters the first iteration of loop.

**Post-Condition**

After the loop termination, the final sum value of total stores the integer for the roman letters from i=0 to i<s.length().

**Loop variant**

The loop must terminate after sum of all integers values of Roman Letter and returns the total.

In other words, loop should not terminate until the integer values of all roman letters sum.

i.e. i<=length()-1 or i<s.length()

**Loop invariant**

Let us assume invariant is true after (i-1)th iteration.

i.e. total stores the integers value of all charAt(i) from i to i-1

We need to design instruction so that the invariant must be after ith iteration of the loop.

i.e. total must be equal to total integer values of all charAt(i) from i=0 to i=n-1

**Steps for the ith iteration**

check the condition while entering the loop i<s.length() true means the value at i stores in the variable s1, i.e. s1=value(s.charAt(i))

while passing the ith value it checks the char and returns the integer value to s1.

And then again it checks the condition for i+1th value which stores in the variable s2, i.e. if(i+1<s.length()) then s2=value(s.charAt(i))

while passing the i+1th value it checks the char and returns the integer value to s2.

After return the integers values to variables s1 and s2 it checks the condition, i.e. if(s1>=s2)

Once the condition satisfies it adds the value and stores it in total, i.e. total=total+s1

False means subtract the s1, i.e. total=total-1

After the inner if condition exits, even if the outer condition also not satisfies it directly adds the integer value of s1 to total.

Otherwise, ignore it and we should increase i by 1 at each iteration the process continues and finally returns total.

**Pseudocode**

int value(char r) {

if (r == 'I' || r=='i')

return 1;

if (r == 'V' || r=='v')

return 5;

if (r == 'X' || r=='x')

return 10;

if (r == 'L' || r=='l')

return 50;

if (r == 'C' || r=='c')

return 100;

if (r == 'D' || r=='d')

return 500;

if (r == 'M' || r=='m')

return 1000;

return -1;

}

int convertRomanToInt(String s) {

int total = 0;

for (int i = 0; i < s.length(); i++) {

int s1 = value(s.charAt(i));

if (i+1 < s.length()) {

int s2 = value(s.charAt(i + 1));

if (s1 >= s2) {

total = total + s1;

} else {

total = total - s1;

}

} else {

total = total + s1;

}

}

return total;

}

**Exercise#3: Move zeros to end of an array**

**Pre-condition**

Define variables: len, i, j, temp

A variable 'i' that acts as a loop counter which is to perform operation for n times

variable 'j' acts as loop counter for finding the zero elements, that executes until matches found

variable "len" acts as condition which loop executes until n becomes false

variable "temp" is used for shifting or interchanging the values

**Post-Condition**

After the loop termination, the zero values must be shifted to end of an array from a[n-1] to a[j]

**Loop variant**

The loop must terminate after shifting all zeros to the end from a[len-1] to a[j]

In other words, loop should not terminate until the shifting process ends

i.e. i<=len-1 or i<n and j<len

**Loop invariant**

Let us assume loop invariant is true after (i-1)th and jth iteration.

i.e. a[j]==0 for the jth iteration

we need design the instruction so that the invariant must be true for jth iteration and ith iteration of loop

i.e. a[n-1]=a[j] inter-changes the value for each iteration len--

**Steps for the ith iteration and jth iteration**

check the condition with arr[i]. if(a[j] == 0), it means we found the zeros of all values from a[j] to a[n-1].

In such situation, interchange the a[j] value to a[n-1] using temp,

i.e. temp = a[j]

a[j]=a[len - 1]

a[len - 1] = temp

After interchanging reduce the length of the array whenever the a[j] == 0

i.e. a[n-1]=a[j] then len--

Otherwise, ignore it and continue the loop from a[n] to a[i]

we should increase i by 1 at each iteration after shifting values from a[n-1] to a[i]

**Pseudocode**

int len = a.length;

for (int i = 0; i < len; i++) {

for (int j = i; j < len; j++) {

if (a[j] == 0) {

int temp = a[j];

a[j] = a[len - 1];

a[len - 1] = temp;

}

}

len--;

}

**Exercise#4: Some mistakes in loop programming**

**Infinite loop Examples**

**Example 1:**

void infiteloop(int i) {

while ( i != 0) {

i = i - 1

}

}

For i<0 , this goes for infinite loop!

**Example 2:**

int i = 0

while(1) {

i = i + 1

print(i)

}

Here loop condition is “1” which is always true!