5 સમસ્યા ઉકેલ : ફ્લોચાર્ટ અને અલ્ગોરિધમ

પરિચય

પથ્થર યુગથી લઈને આધુનિક ટેક્નોલોજીના યુગ સુધી, મનુષ્યે જીવનભર દરરોજ વિવિધ પ્રકારની સમસ્યાઓનો સામનો કરવો પડ્યો છે. આમાં જુદી જુદી વસ્તુઓની ગણતરી કરવી, જુદા જુદા ભાવની અને જુદા જુદા જથ્થામાં ખરીદેલી વસ્તુઓના બિલનો સરવાળો કરવો જેવા ખૂબ જ સરળ કાર્યોથી માંડીને ઘરવપરાશની વસ્તુઓ બનાવતી એક મોટી ફેક્ટરીના સ્ટોરમાં સેંકડો વસ્તુઓના સ્ટોકનું સંચાલન કરવા જેવી જટિલ સમસ્યાઓનો સમાવેશ થાય છે. લોકો તેમના રોજિંદા અને સરળ પ્રશ્નોને કેવી રીતે ઉકેલવા તે માટે ટેવાયેલા છે. પરંતુ જ્યારે મોટી અને જટિલ સમસ્યાઓની વાત આવે, જ્યાં સમયસર અને ચોકસાઈ સાથે કાર્ય પૂર્ણ કરવું ફરજિયાત હોય, ત્યારે કમ્પ્યૂટર મદદ માટે આવે છે.

કમ્પ્યૂટર વિજ્ઞાન આપણને સમસ્યાઓ ઉકેલવાની અનુકૂળ રીત પૂરી પાડે છે અથવા આપણે કહી શકીએ કે સમસ્યાનો ઉકેલ (નિરાકરણ) એ કમ્પ્યૂટર વિજ્ઞાનના કેન્દ્રમાં છે. આપણે જે કામ કરવા માંગીએ છીએ તેના માટે પ્રોગ્રામ લખીને કરી શકાય છે. એકવાર જે તે કામ માટે પ્રોગ્રામ લખાઈ જાય, પછી તે સમસ્યાને વારંવાર ઉકેલ્યા વિના લાંબા સમય સુધી તેને વાપરી શકાય છે. ઉદાહરણ તરીકેઃ બેંકિંગ કામગીરી કરવી, ઓનલાઈન વસ્તુની પસંદગી કરવી અથવા ઓનલાઈન વસ્તુઓનો ઓર્ડર આપવો અને UPI (Unified Payment Interface – યુનીફાઈડ પેમેન્ટ ઈન્ટરફેસ)નો ઉપયોગ કરીને ઓનલાઈન ચુકવણી કરવી.

આ પ્રકરણમાં, આપણે સમસ્યા ઉકેલના ખ્યાલો સમજવા પર અને ત્યારબાદ બે વ્યાપકપણે ઉપયોગમાં લેવાતી પદ્ધતિઓ, (1) ફ્લોચાર્ટ અને (2) અલ્ગોરિધમ – ને સંખ્યાબંધ ઉદાહરણો સાથે શીખવા પર ધ્યાન કેન્દ્રિત કરીશું. આ પ્રકરણ શીખ્યા પછી, વિદ્યાર્થીઓ આપવામાં આવેલ સમસ્યા કે પ્રશ્ન સમજી શકશે અને સમસ્યાને સ્પષ્ટતા સાથે ઉકેલવા માટે ફ્લોચાર્ટ બનાવી શકશે અને/અથવા એલ્ગોરિધમ લખી શકશે.

સમસ્યા ઉકેલની સામાન્ય સમજ (Overview of Problem Solving)

સમસ્યા ઉકેલએ કોઈપણ સમસ્યાનું ઉકેલ શોધવા માટે અપનાવાતી ક્રમબદ્ધ પગલાવાર પ્રક્રિયા છે. તેનો અર્થ છે કે, સમસ્યાને સંપૂર્ણપણે સમજવી અને તેને નાના પગલાઓમાં યોગ્ય ક્રમમાં ગોઠવવી. સમસ્યા ઉકેલના મુખ્ય પગલાં નીચે મુજબ હોઈ શકે છેઃ

- સમસ્યાને તમામ વિગતો સાથે વ્યાખ્યાયિત કરવી
- સમસ્યાને સમજવી
- ઇનપુટ્સ ઓળખવા અથવા મેળવવા
- સમસ્યા હલ કરવા માટે જરૂરી પગલાં લેવા
- આઉટપુટનું પરીક્ષણ કરવું અને માન્ય કરવા

"સમસ્યા હલ કરવા માટે જરૂરી પગલાં લેવા" પગલું સમસ્યા પ્રમાણે બદલાય છે. સરળ સમસ્યાઓ માટે તેમાં ફક્ત થોડા પગલાં જ ક્રમ પ્રમાણે હોય છે. પરંતુ મધ્યમથી મોટી સમસ્યાઓ માટે તેમાં નિર્ણયો લેવાના પગલાં સામેલ થાય છે જેમાં વૈકલ્પિક માર્ગ અપનાવવા પડે, અથવા કેટલાક પગલાં વારંવાર પુનરાવર્તિત કરવા પડે. તેથી આ તબક્કામાં મુખ્યત્વે નીચેના પગલાં વિવિધ સંયોજન સાથે સામેલ થાય છેઃ

- **ક્રમ (Sequence) :** એક પછી એક પગલું
- **નિર્ણય લેવો (Decision Making) :** બે કે વધુ વિકલ્પમાંથી કોઈ એક પસંદ કરવો
- **પુનરાવર્તન/લૂપિંગ (Repetiition/Looping) :** કેટલાક પગલાં વારંવાર કરવાનું, એટલે કે એકથી વધારે વખત કરવા.

ચાલો હવે ઉદાહરણની મદદથી તેને સમજીએ. બે સંખ્યાનો સરવાળો કરવા માટેના પગલાં નીચે મુજબ છે:

1. બે સંખ્યાઓ N1 અને N2 સ્વીકારો
2. તેમનો સરવાળો કરો: SUM = N1 + N2
3. પરિણામ એટલે કે SUM પ્રિન્ટ કરો

ઉપરોક્ત સમસ્યા સંપૂર્ણપણે ક્રમબદ્ધ છે અને તેને ઉકેલવા માટે આપણે પગલાં 1, 2 અને 3 ને ક્રમ પ્રમાણે અનુસરવા પડે છે.

બે સંખ્યાઓમાંથી મોટી સંખ્યા શોધવાના પગલાં નીચે મુજબ છે:

1. બે સંખ્યાઓ N1 અને N2 સ્વીકારો
2. તેમની સરખામણી કરો, જો N1 > N2 હોય તો N1 મોટી છે, પછી સીધું પગલું-4 પર જાઓ
3. નહીંતર N2 મોટી છે
4. મોટી સંખ્યા પ્રિન્ટ કરો

નિરીક્ષણ કરો કે ઉપરોક્ત સમસ્યામાં પગલું-2 પર નિર્ણય સામેલ છે. પગલું-1 માં N1 અને N2 સ્વીકાર્યા પછી, પગલું-2 પહેલા તેમની ">" દ્વારા સરખામણી કરે છે. જો N1 મોટી હોય તો તે પગલું-3 છોડીને સીધું પગલું-4 પર જઈ N1 પ્રિન્ટ કરે છે. જો સરખામણી ખોટી નીવડે તો પગલું-2 નો બાકીનો ભાગ છોડી દેવામાં આવે છે, એટલે કે N2 મોટી હોવાથી પગલું-3 કરવામાં આવે છે અને ત્યારબાદ પગલું-4 N2 પ્રિન્ટ કરે છે. અહીં મુખ્ય વાત એ છે કે આપણે N1 અને N2ની તુલનાના આધારે બે વિકલ્પોમાંથી એક પસંદ કરીએ છીએ.

1 થી 10 સુધીના સંખ્યાઓનો સરવાળો કરવા માટેના પગલાં નીચે મુજબ છે:

1. SUM = 0 થી પ્રારંભ કરો
2. I = 1 નક્કી કરો
3. પગલું-4 અને પગલું-5 ને 10 વખત પુનરાવર્તિત કરો
4. SUM માં I ઉમેરો
5. I ને 1 થી વધારો
6. SUM પ્રિન્ટ કરો

ઉપરોક્ત સમસ્યાના પગલાં ધ્યાનથી જુઓ. પગલું-1માં SUM ને 0 થી શરૂ કરવામાં આવે છે અને પગલું-2માં I ને પ્રથમ મૂલ્ય 1 આપવામાં આવે છે. ત્યારબાદ પગલું-4માં I ની વર્તમાન મૂલ્ય SUM સાથે ઉમેરવામાં આવે છે અને પગલું-5માં I ને 1થી વધારો કરવામાં આવે છે. આ પ્રક્રિયા કુલ 10 વાર થાય છે, એટલે કે I ના મૂલ્યો ક્રમશઃ 1, 2, 3, 5, ..., 10 થાય છે અને દરેક વખતે તે SUM સાથે ઉમેરાય છે. આ રીતે અંતે SUM = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55 મળે છે, જેને પગલું-6 દ્વારા પ્રિન્ટ કરવામાં છે.

ઉપરોક્ત ઉદાહરણમાં સમસ્યા ઉકેલવાના પ્રક્રિયા પગલાંના ત્રણેય મહત્વના ઘટકોનો ઉપયોગ દર્શાવવામાં આવ્યો છે. વાસ્તવિક જીવનમાં આપણી સમક્ષ આવતી સમસ્યાઓ ઉપરના ઉદાહરણ જેટલી સરળ નથી. તેમાં અનેક ગણતરીઓ સામેલ હોય છે અને તે જટિલ હોય છે. આવી સમસ્યાઓ ઉકેલવા માટે આપણે ઉપર જણાવેલ ત્રણ પગલાં - ક્રમ (Sequence), નિર્ણય લેવો (Decision Making) અને પુનરાવર્તન (Repetiition) - નો વિવિધ સંયોજનમાં ઉપયોગ કરવો પડે છે, જે સમસ્યાની જરૂરિયાત પર આધારિત હોય છે.

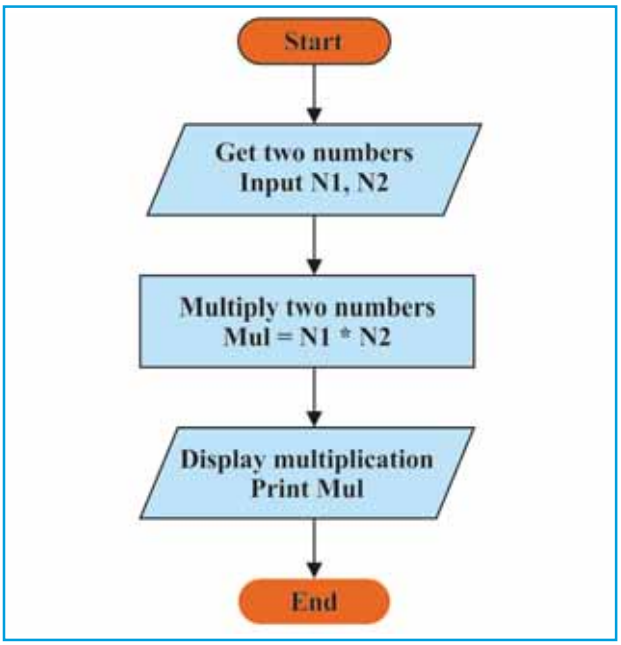
આગળનો વિભાગ તમને બે વ્યાપકપણે વપરાતા સમસ્યા ઉકેલવાના ઉપાયો - ફ્લોચાર્ટ અને અલ્ગોરિધમ - સાથે પરિચિત કરાવશે. આપણે સમજીશું કે કેવી રીતે આ બંનેનો ઉપયોગ કરીને ઘણી ગણતરીઓ ધરાવતી સમસ્યાઓ ઉકેલી શકાય અને અંતે ફ્લોચાર્ટ તથા અલ્ગોરિધમની તુલના પણ કરીશું.

ફ્લોચાર્ટ અને તેના પ્રતીકો (Flowchart and its Symbols)

ચાલો, આપણે ફ્લોચાર્ટને વિગતવાર સમજીએ, જેમાં ફ્લોચાર્ટ બનાવવા માટે ઉપયોગમાં લેવાતા વિવિધ પ્રતીકો તેમના અર્થ અને ઉપયોગનો સમાવેશ થાય છે.

ફ્લોચાર્ટ (Flowchart)

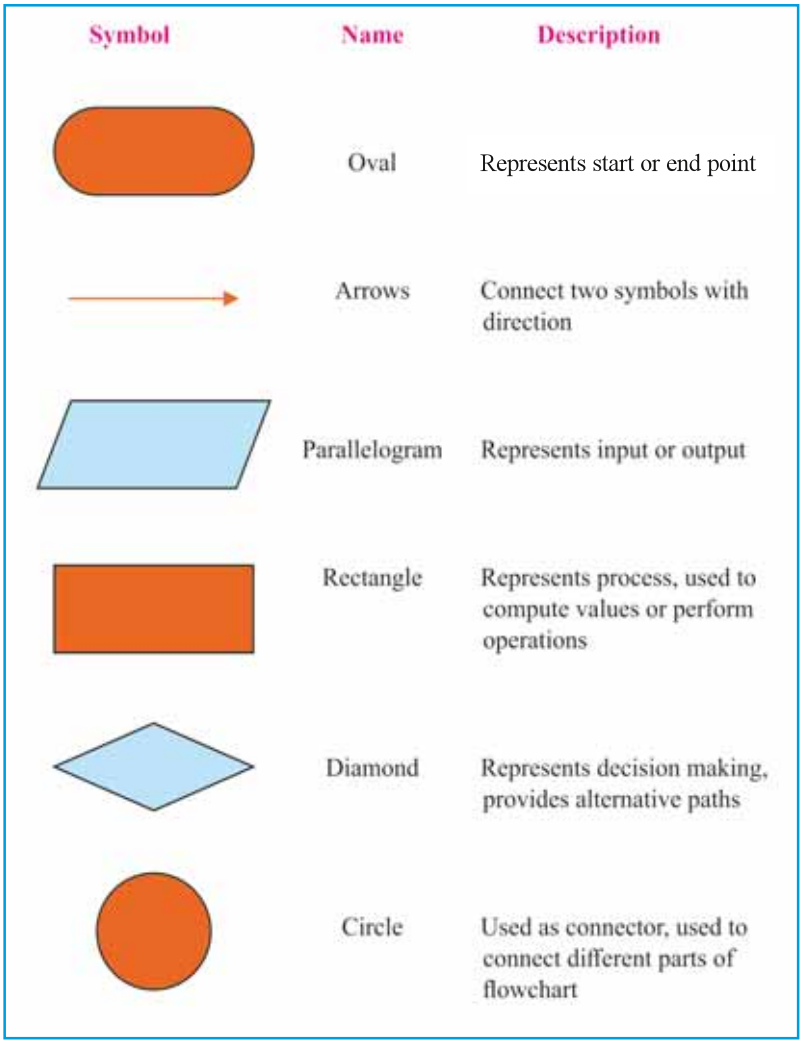
ફ્લોચાર્ટ એ સમસ્યાના ઉકેલનું ચિત્રાત્મક નિરૂપણ છે. તે કોઈ પ્રક્રિયા અથવા કાર્યના પગલાંને સરળ અને દ્રશ્યમાન રીતે દર્શાવે છે. ફ્લોચાર્ટમાં જુદા જુદા પગલાં માટેની વિવિધ ક્રિયાઓ દર્શાવવા માટે અંડાકાર (ovals), લંબચોરસ (rectangles) અને ડાયમંડ (diamonds) જેવા વિવિધ પ્રતીકોનો ઉપયોગ થાય છે. વિવિધ પગલાંને જોડવા માટે એરોનો ઉપયોગ થાય છે. ફ્લોચાર્ટ દ્રશ્ય નિરૂપણનો ઉપયોગ કરતો હોવાથી, તમામ સંભવિત માર્ગો અને પગલાં સ્પષ્ટપણે દેખાય છે, અને તે ઉકેલની સમજણ ખૂબ જ સરળ બનાવે છે. આકૃતિ 5.1 બે સંખ્યાઓનો ગુણાકાર કરવા માટેનો ફ્લોચાર્ટ દર્શાવે છે. આકૃતિ 5.1માં દર્શાવ્યા મુજબ, ફ્લોચાર્ટ "Start" (શરૂઆત) થી શરૂ થાય છે અને "End" (અંત)થી સમાપ્ત થાય છે. બીજું પગલું, જે સમાંતર ચતુષ્કોણ (parallelogram) દ્વારા દર્શાવવામાં આવ્યું છે, તે બે સંખ્યાઓ N1 અને N2 સ્વીકારે છે. ત્રીજું પગલું, લંબચોરસનો ઉપયોગ કરીને તેમનો ગુણાકાર કરવાની પ્રક્રિયા દર્શાવે છે : Mul = N1 * N2. ચોથું પગલું ફરીથી સમાંતર ચતુષ્કોણ છે, જે આઉટપુટ એટલે કે બે આપેલી સંખ્યાઓનો ગુણાકાર (Mul) પ્રિન્ટ કરવા માટે છે. આ ફ્લોચાર્ટ ખૂબ જ સરળ છે કારણ કે સમસ્યા ક્રમિક છે. પ્રવાહ એટલે કે પગલાંનો ક્રમ સ્પષ્ટપણે દેખાય છે, કારણ કે દરેક પગલાં પછીનો એરો આગલા પગલા તરફ નિર્દેશ કરે છે. નિર્ણય લેવા અને પુનરાવર્તન ધરાવતી વિવિધ સમસ્યાઓ માટે ફ્લોચાર્ટ વિકસાવીએ તે પહેલાં, ચાલો આપણે ફ્લોચાર્ટમાં ઉપયોગમાં લેવાતા તમામ પ્રતીકોને સમજીએ.



આકૃતિ 5.1 : ગુણાકાર કરવા માટેનો ફ્લોચાર્ટ

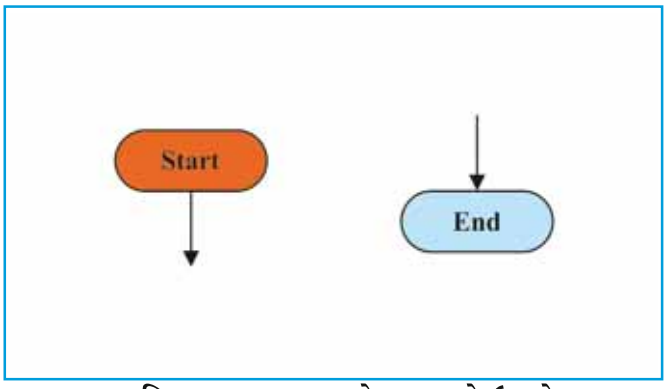
ફ્લોચાર્ટ પ્રતીકો (Flowchart Symbols)

આકૃતિ 5.2માં ફ્લોચાર્ટ બનાવવા માટે ઉપયોગમાં લેવાતા વિવિધ પ્રતીકો દર્શાવેલ છે. આકૃતિમાં તેને સમજવા માટે પ્રતીક, તેનું નામ અને ટૂંકું વર્ણન



આકૃતિ 5.2 : ફ્લોચાર્ટ પ્રતીકો

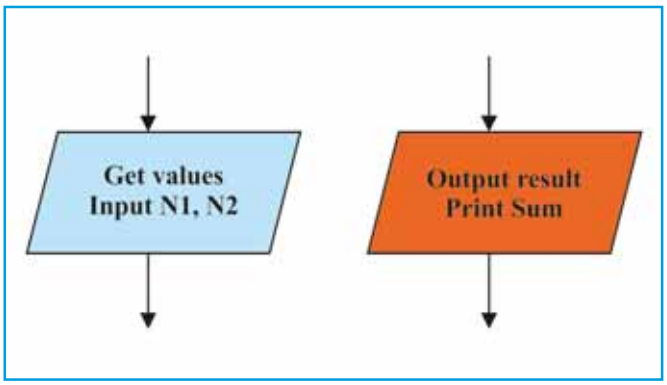
આપેલ છે. ચાલો હવે ફ્લોચાર્ટમાં તેમના ઉપયોગ સાથે તે દરેકને એક પછી એક સમજીએ.



આકૃતિ 5.3 : Start અને End નો ઉપયોગ

શરૂઆત/અંત (Start/End) : Start અને End એ અંડાકાર (oval) દ્વારા રજૂ થતા ટર્મિનલ પ્રતીકો છે, જે અનુક્રમે ફ્લોચાર્ટની શરૂઆત અને અંત દર્શાવે છે. Start ફ્લોચાર્ટની શરૂઆત દર્શાવે છે અને તે હંમેશાં પ્રથમ પ્રતીક હોય છે. End ફ્લોચાર્ટની પૂર્ણતા અથવા સમાપ્તિ દર્શાવે છે અને તે હંમેશાં છેલ્લું પ્રતીક હોય છે. આકૃતિ 5.1માં બતાવ્યા પ્રમાણે, પ્રક્રિયા અથવા કાર્ય કરવાનાં પગલાં Start અને End પ્રતીકોની વચ્ચે સમાવિષ્ટ હોય છે. આકૃતિ 5.3 એરો સાથે Start અને End પ્રતીકના ઉપયોગને દર્શાવે છે.

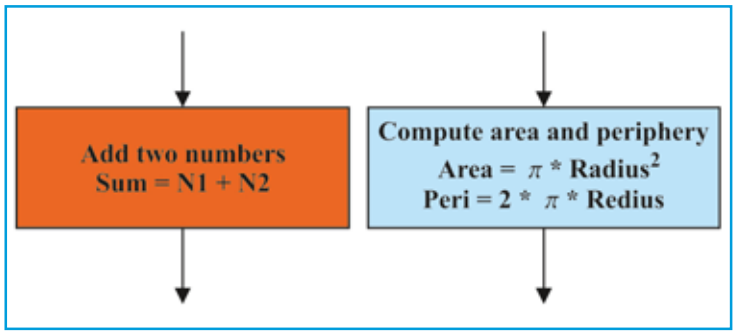
એરો (Arrow) : એરોનો ઉપયોગ પ્રવાહ રેખાઓ દર્શાવવા માટે થાય છે. ફ્લોચાર્ટમાં આગળનું પ્રતીક ક્યાં દોરેલું છે તેના આધારે આપણે તમામ દિશાઓમાં એરોનો ઉપયોગ કરી શકીએ છીએ. આકૃતિ 5.1માં બતાવ્યા પ્રમાણે, સૌથી સામાન્ય ઉપયોગ નીચે તરફનો એરો છે. એરો ફ્લોચાર્ટની અંદર અમલનો માર્ગ નક્કી કરવા માટે વપરાતું સૌથી મહત્વપૂર્ણ પ્રતીક છે, જે સૂચવે છે કે નિયંત્રણ હવે પછી ક્યાં જાય છે. જો કે, ખાસ કરીને એક જ પૃષ્ઠ પર ઘણાં પગલાં ધરાવતા મોટા ફ્લોચાર્ટમાં જરૂર પડે ત્યારે, ડાબી કે જમણી બાજુના એરોનો પણ ઉપયોગ થાય છે.



આકૃતિ 5.4 : ઈનપુટ/આઉટપુટ પ્રતીકોનો ઉપયોગ

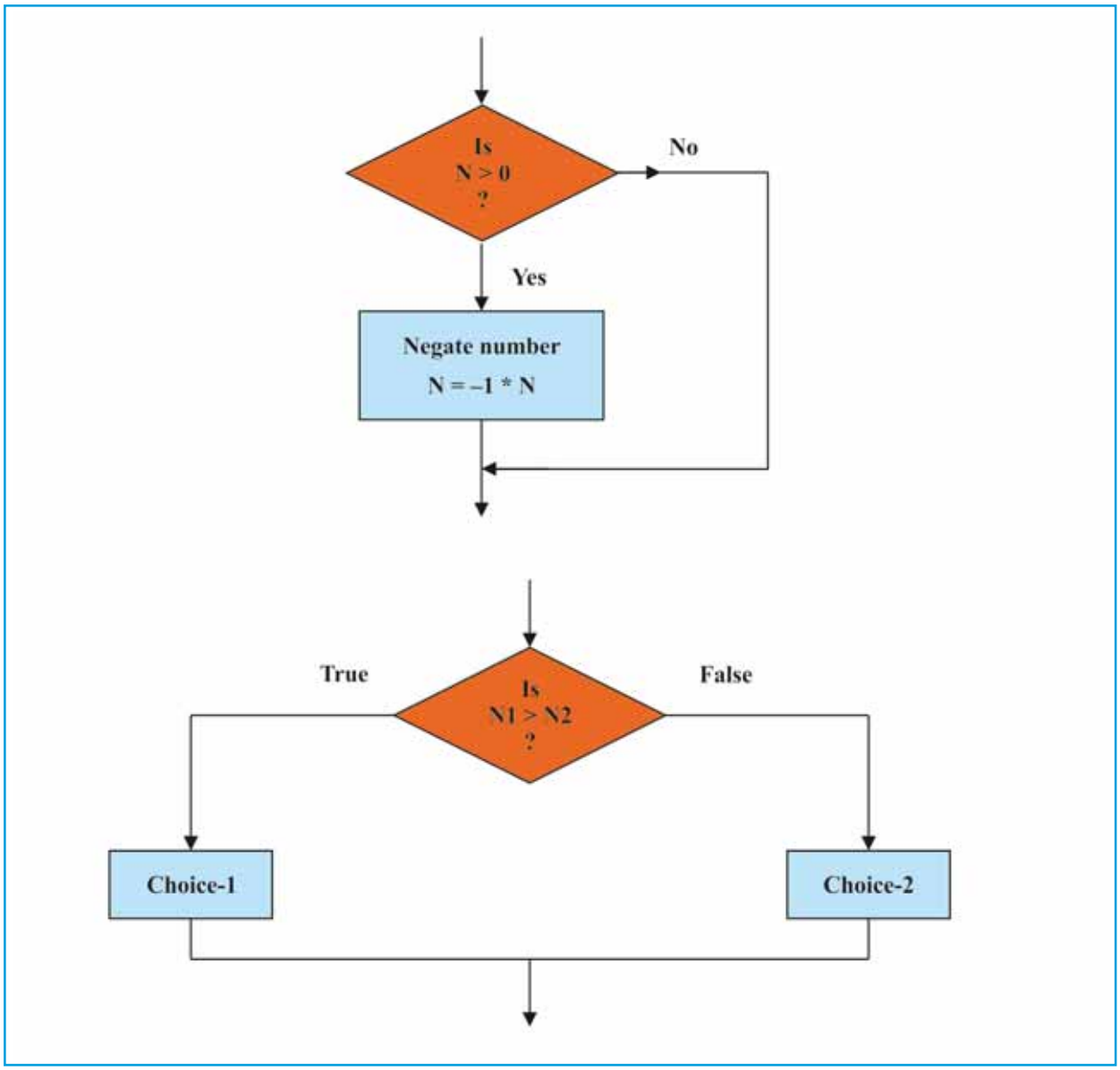
ઈનપુટ/આઉટપુટ (Input/Output) : ઈનપુટ અને આઉટપુટને આકૃતિ 5.2માં બતાવ્યા પ્રમાણે સમાંતર ચતુષ્કોણ દ્વારા રજૂ કરવામાં આવેલ છે. ઈનપુટ પ્રતીકનો ઉપયોગ યુઝર અથવા વાસ્તવિક દુનિયા તરફથી મૂલ્યો અથવા ડેટા મેળવવા માટે થાય છે, જેના પર આગળની પ્રક્રિયા કરવામાં આવે છે. આઉટપુટ પ્રતીક દર્શાવે છે કે ઉકેલ દ્વારા ઉત્પન્ન થયેલું પરિણામ યુઝર અથવા વાસ્તવિક દુનિયાને તેમના ઉપયોગ માટે મોકલવામાં આવે છે. આકૃતિ 5.4 ઈનપુટ અને આઉટપુટ પ્રતીકોનો ઉપયોગ દર્શાવે છે. ઈનપુટ પ્રતીકને "Input N1 and N2" દ્વારા અને આઉટપુટ પ્રતીકને "Print Sum" દ્વારા દર્શાવવામાં આવ્યું છે.

પ્રક્રિયા (Process) : તેના માટે લંબચોરસ (rectangle)નો ઉપયોગ થાય છે. જ્યારે પણ આપણે કોઈ મૂલ્યોની ગણતરી કે પ્રક્રિયા કરવા માંગીએ છીએ, ત્યારે તેનો ઉપયોગ થાય છે. પ્રક્રિયાનું પગલું વિવિધ ગણતરીઓ કરવા માટે વાપરી શકાય છે, જેમ કે અંકગણિતીય ક્રિયાઓ, તાર્કિક ક્રિયાઓ, આપેલા મૂલ્યોના આધારે સૂત્રોનું મૂલ્યાંકન વગેરે. લંબચોરસ દ્વારા રજૂ કરાયેલ એક જ 'પ્રક્રિયા બોક્સ' માં પગલાંના સમૂહનો ઉપયોગ કરીને એક કરતાં વધુ મૂલ્યોની ગણતરી કરી શકાય છે. આકૃતિ 5.5 પ્રક્રિયાના પગલાંના બે ઉદાહરણો દર્શાવે છે: પહેલું બે સંખ્યાઓનો સરવાળો કરવા માટે (Sum = N1 + N2) અને બીજું વર્તુળનું ક્ષેત્રફળ અને પરિઘની ગણતરી કરવા માટે.



આકૃતિ 5.5 : પ્રક્રિયા માટેના પ્રતિકનો ઉપયોગ

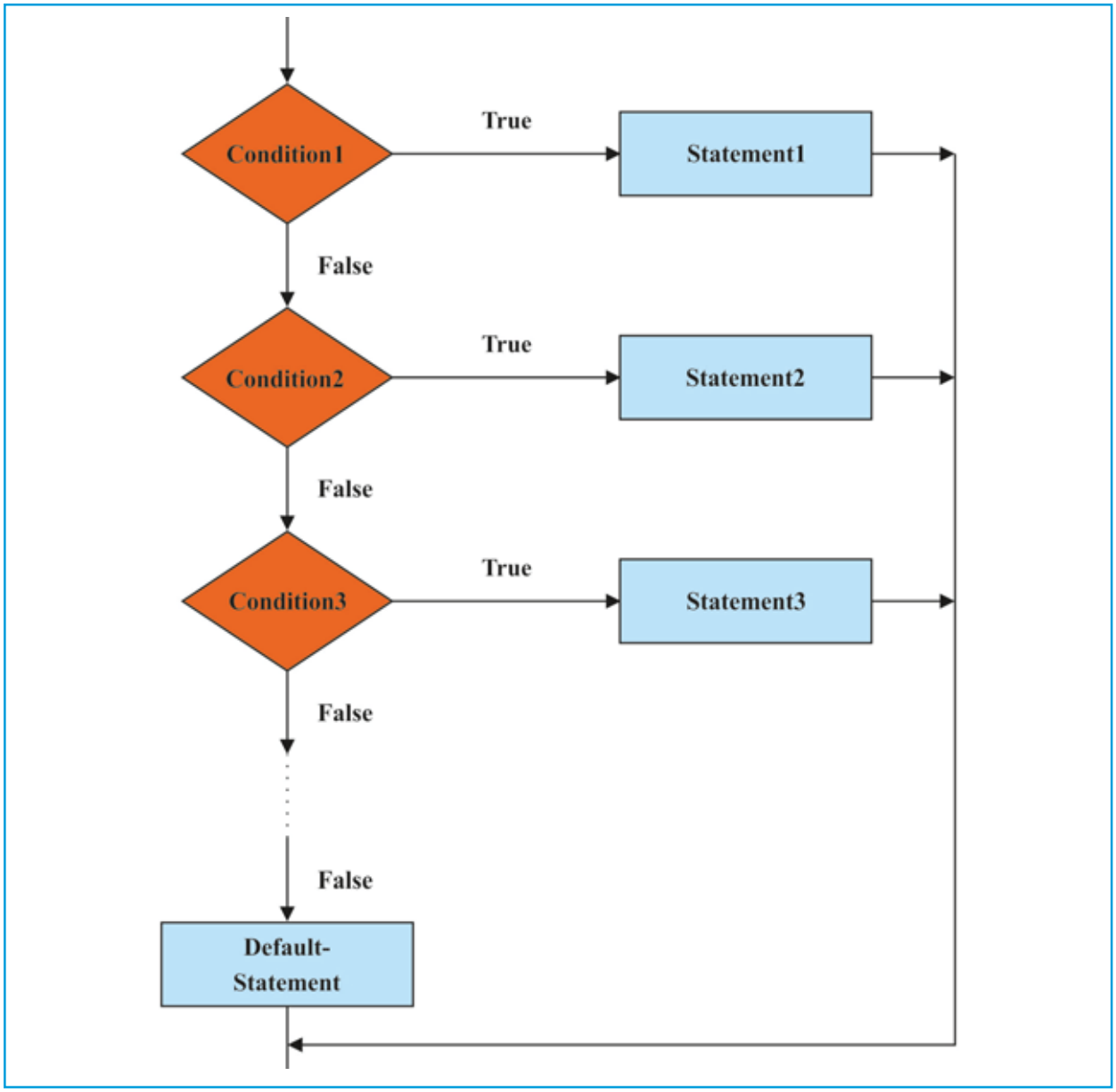
નિર્ણય (Decision) : નિર્ણય લેવો એ સૌથી મહત્વપૂર્ણ છે અને માનવીઓ દ્વારા તેમના જીવનમાં નિયમિતપણે કરવામાં આવે છે. ફ્લોચાર્ટમાં નિર્ણય માટે ડાયમંડ આકારના પ્રતીકનો ઉપયોગ થાય છે. આપણે ડાયમંડની અંદર સામાન્ય રીતે શરતનો ઉપયોગ કરીને વિકલ્પોમાંથી એક નક્કી કરીએ છીએ. ધારો કે, જો સંખ્યા 0 કરતાં મોટી હોય તો આપણે તેને ઋણ સંખ્યા બનાવીએ છીએ, નહીં તો કોઈ પગલું લેતા નથી. બીજું ઉદાહરણ બે સંખ્યાઓની તુલના કરે છે અને જો પ્રથમ સંખ્યા બીજી સંખ્યા કરતાં મોટી હોય તો આપણે વિકલ્પ-1 (choice-1) લઈએ છીએ, નહીં તો આપણે વિકલ્પ-2 (choice-2) લઈએ છીએ. આકૃતિ 5.6માં આ દર્શાવ્યું છે.



આકૃતિ 5.6 : નિર્ણય માટેના પ્રતિકનો ઉપયોગ

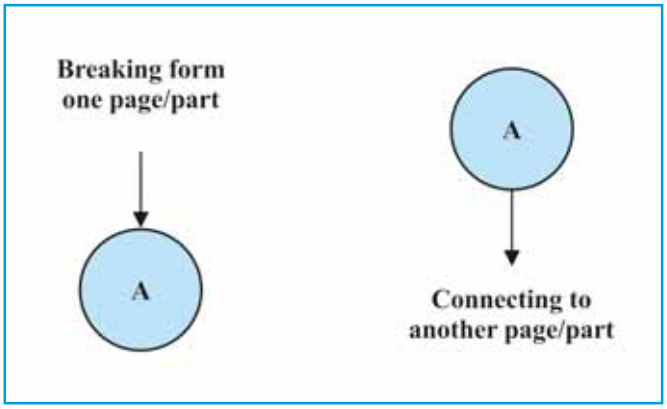
કેટલીકવાર, આપણી પાસે બહુવિધ શરતો હોય છે અને તે દરેક સાથે અમુક વિધાનો (statements) જોડાયેલા હોય છે. આકૃતિ 5.7 આવી પરિસ્થિતિ દર્શાવે છે. આ એક મલ્ટિ-ચોઈસ માળખું છે, કારણ કે કઈ શરત સાચી છે તેના આધારે આપણે એક કરતાં વધારે વિકલ્પોમાંથી એક વિકલ્પ પસંદ કરવો પડે છે. આકૃતિ 5.7માં, સૌપ્રથમ શરત-1 (Condition1) નું પરીક્ષણ થાય છે. જો તે સાચી હોય, તો વિધાન-1 (statement1) ની પ્રક્રિયા થાય છે અને નિયંત્રણ માળખાના અંત તરફ આગળ વધે છે. જો તે ખોટી હોય, તો પછી શરત-2 (Condition2) નું મૂલ્યાંકન થાય છે. જો શરત-2 (Condition2) સાચી હોય તો વિધાન-2 (statement2) ની પ્રક્રિયા થાય છે અને તે અંત તરફ આગળ વધે છે, અન્યથા તે શરત-3 (Condition3) તરફ જાય છે. જ્યાં સુધી બધી શરતોનું

પરીક્ષણ ન થાય ત્યાં સુધી આ પ્રક્રિયા ચાલુ રહે છે. જો બધી શરતો ખોટી હોય તો પછી ડિફોલ્ટ-વિધાન (Default-statement) ની પ્રક્રિયા થાય છે અને માળખું પૂર્ણ થાય છે. ટૂંકમાં, કઈ શરત સાચી છે તેના આધારે ફક્ત એક જ વિધાન અમલમાં મૂકવામાં આવશે અથવા ડિફોલ્ટ-વિધાનનો અમલ થશે.



આકૃતિ 5.7 : નિર્ણય દ્વારા મલ્ટિ-ચોઈસનો ઉપયોગ

કનેક્ટર (Connector) : સિંગલ કેપિટલ અક્ષર ધરાવતું નાનું વર્તુળ કનેક્ટર તરીકે વપરાય છે. જ્યારે કોઈ સમસ્યા પ્રમાણમાં મોટી અને જટિલ હોય છે, ત્યારે ફ્લોચાર્ટ પણ જટિલ બની જાય છે અથવા એક જ પૃષ્ઠમાં સમાઈ શકતો નથી. કનેક્ટર્સનો ઉપયોગ ફ્લોચાર્ટના બે ભાગોને એક જ પૃષ્ઠમાં અથવા અન્ય પૃષ્ઠમાં જોડવા માટે થાય છે. કનેક્ટરનો ઉપયોગ હંમેશાં જોડીમાં થાય છે: ધારો કે, "A" અક્ષર ધરાવતું એક વર્તુળ અંદર તરફ આવતા એરો સાથે (કોઈ



આકૃતિ 5.8 : કનેક્ટર્સ

અન્ય પ્રતીકમાંથી બહાર આવતું) બતાવે છે કે ફ્લોચાર્ટ તે જ પૃષ્ઠના અન્ય ભાગમાં અથવા અન્ય પૃષ્ઠમાં ચાલુ રહે છે. આ જ "A" અક્ષર ધરાવતું બીજું વર્તુળ બહાર તરફ જતા એરો સાથે તે જ પૃષ્ઠમાં અથવા અન્ય પૃષ્ઠમાં ફ્લોચાર્ટના બીજા ભાગ સાથે જોડાય છે. આકૃતિ 5.8માં "A" અક્ષર સાથેના કનેક્ટર્સની જોડી દર્શાવેલ છે.

ક્રમિક અને નિર્ણય લેવાના ઉદાહરણો (Sequential and Decision-making Examples)

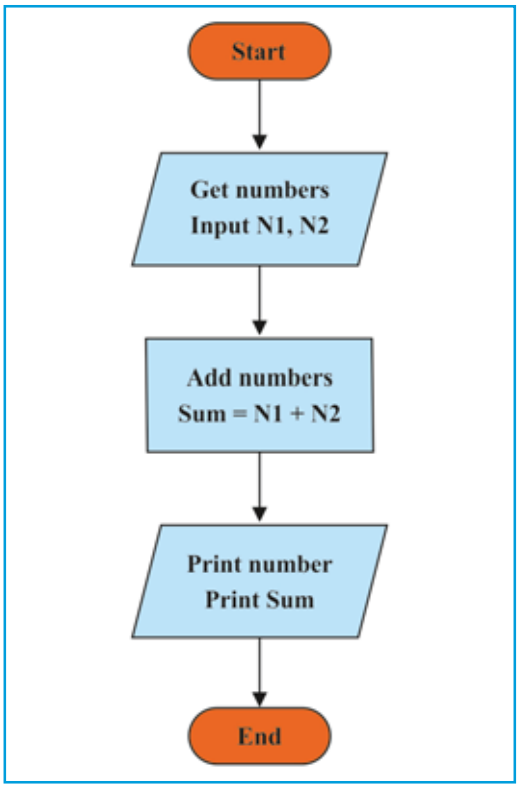
ચાલો હવે અગાઉના વિભાગમાં શીખેલા પ્રતીકોનો ઉપયોગ કરીને સરળ ક્રમિક પગલાં અને નિર્ણય લેવા સહિતની વિવિધ સમસ્યાઓ માટે ફ્લોચાર્ટ વિકસાવીએ.

પ્રશ્ન 1 : આપેલ બે સંખ્યાઓનો સરવાળો કરવા માટે ફ્લોચાર્ટ દોરો.

ઉકેલ :

સમસ્યા ખૂબ જ સરળ છે અને તેમાં બે સંખ્યાઓ મેળવવાની, તેમનો સરવાળો કરવાની અને અંતે તેમનો સરવાળો દર્શાવવાની જરૂર છે. આકૃતિ 5.9 ફ્લોચાર્ટ દર્શાવે છે.

આકૃતિ 5.9માં બતાવ્યા પ્રમાણે, ફ્લોચાર્ટ Start પ્રતીકથી શરૂ થાય છે. પછી તે N1 અને N2 ચલોનો ઉપયોગ કરીને બે સંખ્યાઓ મેળવે છે. ત્રીજું પગલું પ્રક્રિયા છે, જેને લંબચોરસ દ્વારા દર્શાવવામાં આવ્યું છે, જે N1 અને N2 નો સરવાળો કરીને તેને ચલ Sum માં સંગ્રહિત કરે છે. ચોથું પગલું Sum માં સંગ્રહિત સરવાળાને પ્રિન્ટ કરે છે. છેલ્લું પગલું End ફ્લોચાર્ટને સમાપ્ત કરવા માટે છે.



આકૃતિ 5.9 : બે સંખ્યાનાં સરવાળાનો ફ્લોચાર્ટ

આકૃતિ 5.9માં નાના ફેરફારો કરીને આપણે બાદબાકી માટેનો ફ્લોચાર્ટ પણ સરળતાથી દોરી શકીએ છીએ. પ્રથમ ફેરફાર, ત્રીજા પગલાં (પ્રક્રિયા પગલું) ને બદલીને

Diff = N1 – N2

કરવું. અને પછી ચોથા પગલાંને બદલીને નીચે પ્રમાણે કરો.

output the difference

Print Diff

પ્રશ્ન 2 : સાદું વ્યાજ ગણવા નીચેના સૂત્રનો ઉપયોગ કરી ફ્લોચાર્ટ દોરો.

$I = (P \times R \times N)/100$

જ્યાં P = મુદ્દલની રકમ, R = વ્યાજનો દર અને N = સમયગાળો દર્શાવે છે.

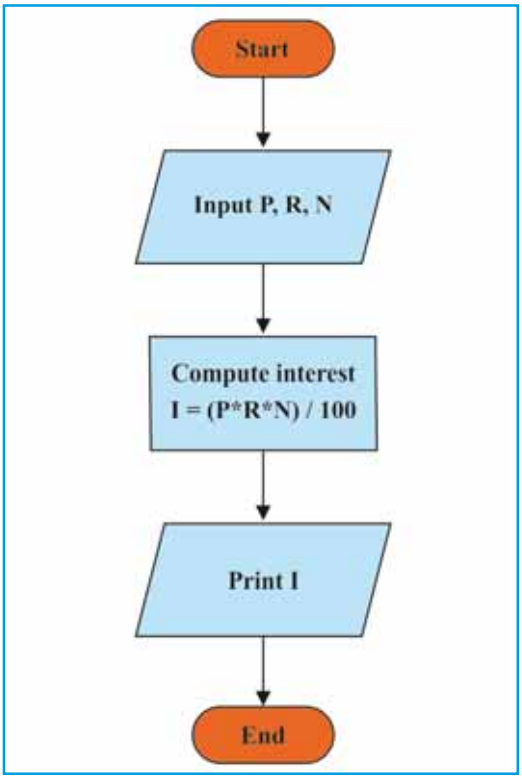
ઉકેલ :

ચાલો એક ઉદાહરણ લઈએ જ્યાં

P = 100000, R = 7% અને N = 1 વર્ષ.

તો વ્યાજ (Interest) I = (100000 * 7 * 1) / 100 = 7000.

આકૃતિ 5.10 ફ્લોચાર્ટ દર્શાવે છે. ફ્લોચાર્ટમાં જોઈ શકાય છે કે ત્રણ ચલો P, R અને N નો ઉપયોગ ઇનપુટ મેળવવા માટે થાય



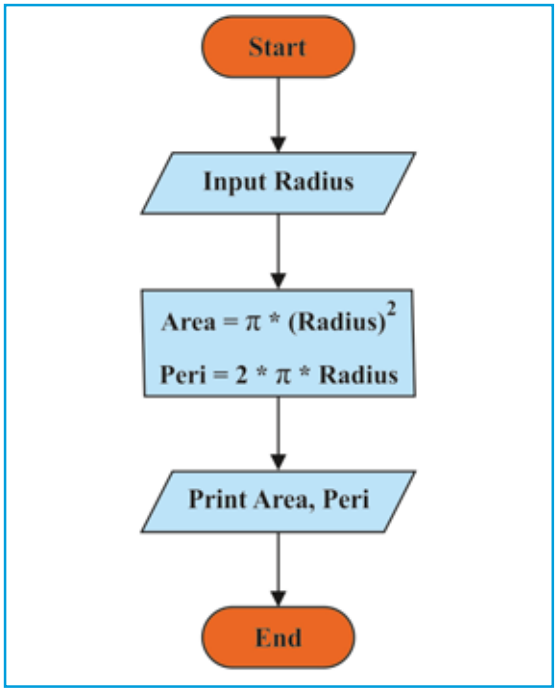
આકૃતિ 5.10 : સાદું વ્યાજ ગણવાનો ફ્લોચાર્ટ

છે, અને પછી પ્રોસેસ બોક્સ ઉપર દર્શાવેલા સૂત્રનો ઉપયોગ કરીને વ્યાજ I ની ગણતરી કરે છે, અને અંતે ગણતરી કરેલ વ્યાજ પ્રિન્ટ કરવામાં આવે છે.

પ્રશ્ન 3 : વર્તુળના ક્ષેત્રફળ અને પરિઘ શોધવાનો ફ્લોચાર્ટ દોરો.

ઉકેલ :

વર્તુળનું ક્ષેત્રફળ અને પરિઘ ગણવા માટે, આપણે ત્રિજ્યાને ઇનપુટ તરીકે સ્વીકારવી પડે. આપણે ક્ષેત્રફળ $A = \pi R^2$ અને પરિઘ $P = 2\pi R$ નો ઉપયોગ કરીને ગણી શકીએ. ફ્લોચાર્ટ આકૃતિ 5.11માં દર્શાવેલ છે. ધ્યાન આપો કે આપણે એક જ પ્રોસેસિંગ બોક્ષમાં બે મૂલ્યોની ગણતરી કરી રહ્યા છીએ. જો તેઓ ક્રમમાં હોય તો આપણે એક જ પ્રોસેસ બોક્સમાં ગમે તેટલા મૂલ્યોની ગણતરી કરી શકીએ છીએ.

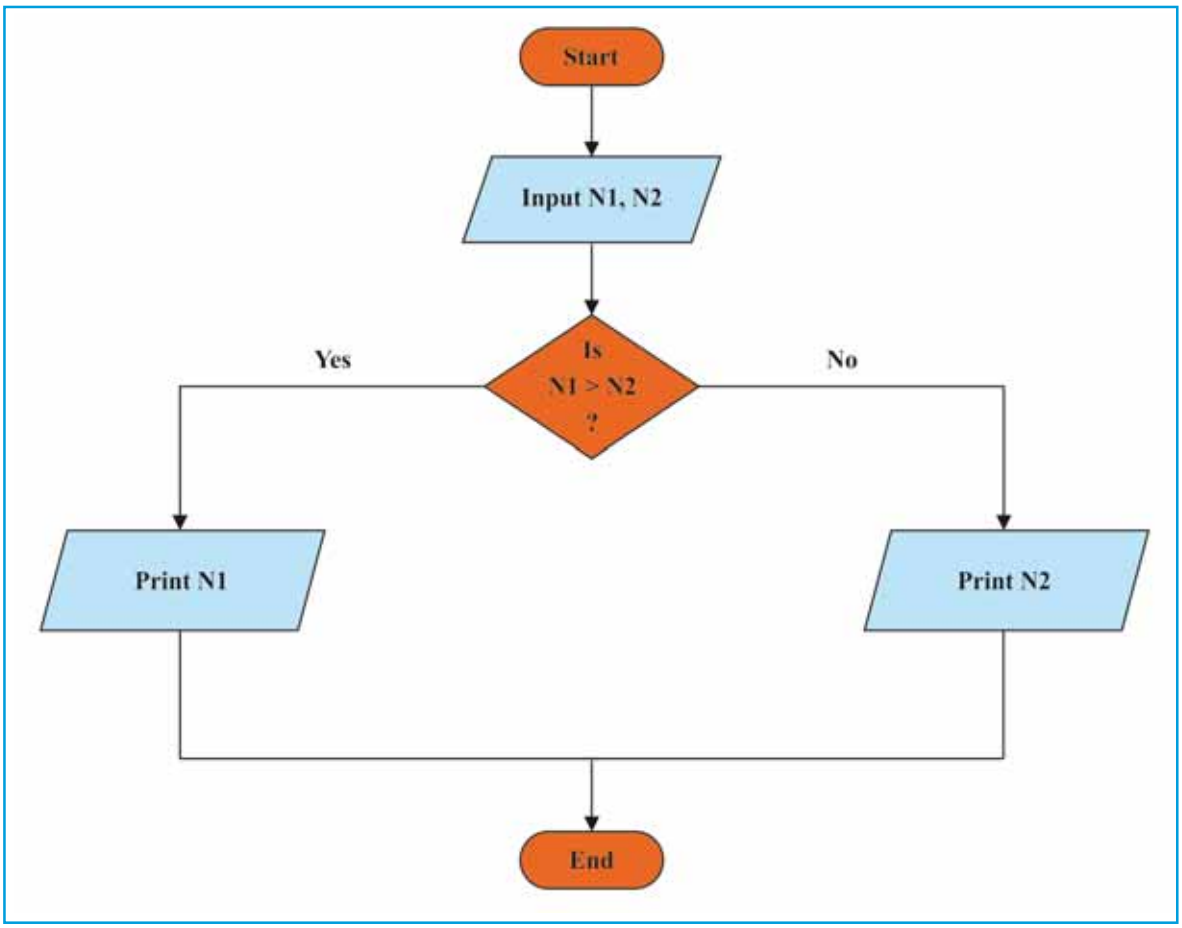


આકૃતિ 5.11 : વર્તુળના ક્ષેત્રફળ અને પરિઘ શોધવાનો ફ્લોચાર્ટ

પ્રશ્ન 4 : બે સંખ્યામાંથી મોટી સંખ્યા શોધવાનો ફ્લોચાર્ટ દોરો.

ઉકેલ :

આ ફ્લોચાર્ટમાં બે સંખ્યાઓની સરખામણી કરવા માટે ડાયમંડ પ્રતીકનો ઉપયોગ થાય છે, જેમાં ">" (કરતાં મોટું) ચિહ્નનો ઉપયોગ કરવામાં આવે છે અને પછી તેમાંથી જે સંખ્યા મોટી હોય તેને પ્રિન્ટ કરવામાં આવે છે. આ ફ્લોચાર્ટ આકૃતિ 5.12માં દર્શાવેલ છે.



આકૃતિ 5.12 : બે સંખ્યામાંથી મોટી સંખ્યા શોધવાનો ફ્લોચાર્ટ

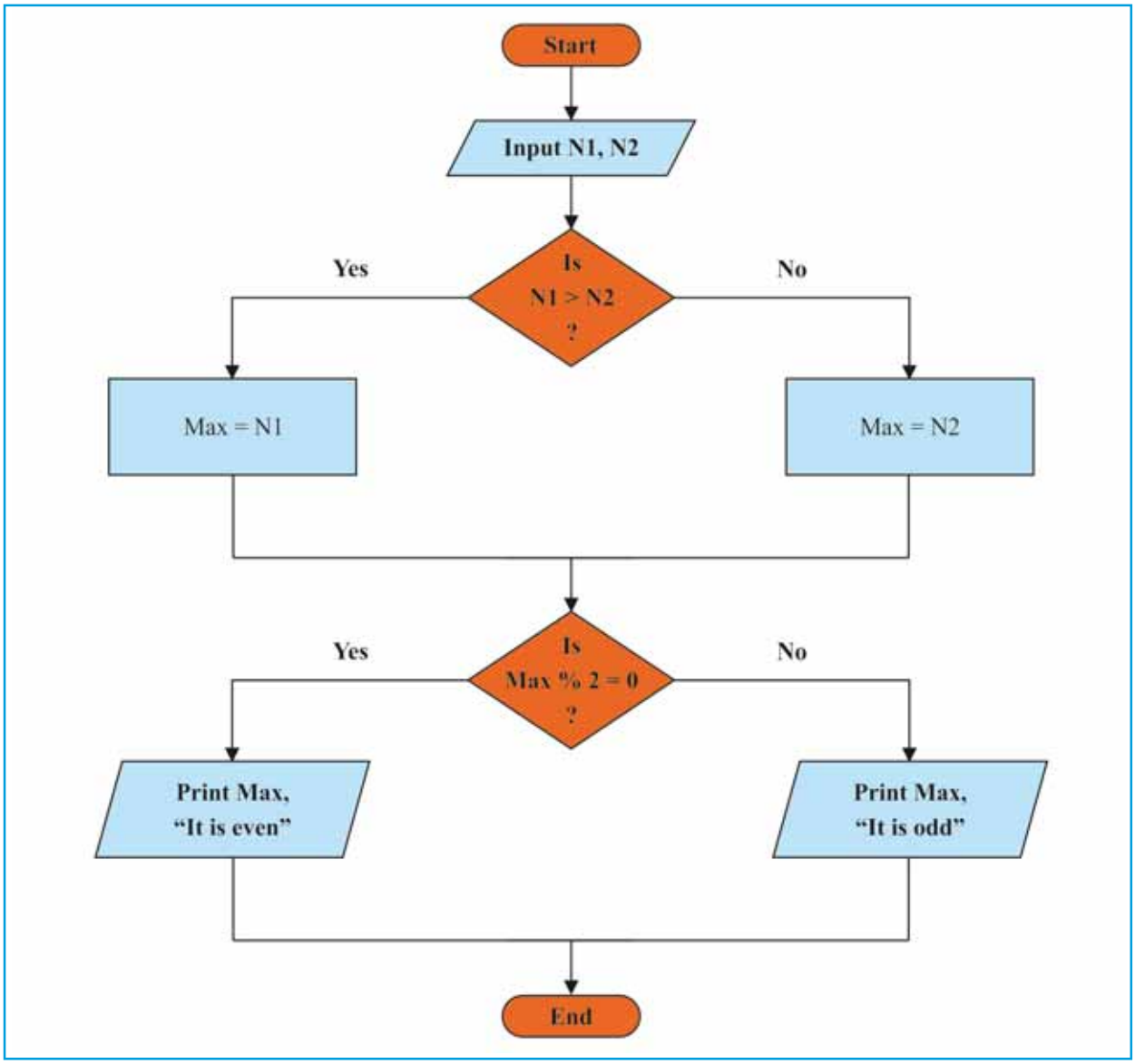
ઉપરના ફ્લોચાર્ટમાં બતાવ્યા પ્રમાણે, N1 અને N2 એમ બે સંખ્યાઓ મેળવ્યા પછી, નિર્ણય બોક્સમાં N1 > N2 શરતનો ઉપયોગ કરીને તેમની સરખામણી કરવામાં આવે છે. નિર્ણય બોક્સ બે માર્ગો બનાવે છે : એક જો શરત સાચી હોય (N1 પ્રિન્ટ કરવું) અને બીજો જો શરત ખોટી હોય (N2 પ્રિન્ટ કરવું). તેમાંથી એક પ્રિન્ટ કર્યા પછી ફ્લોચાર્ટ સમાપ્ત થાય છે.

જો આપણે ફ્લોચાર્ટ આકૃતિ 5.12ના નિર્ણય બોક્સમાં શરતને ફક્ત N1 < N2 માં બદલીએ, તો તે N1 અને N2 માંથી નાની સંખ્યાને પ્રિન્ટ કરશે.

પ્રશ્ન 5 : બે સંખ્યાઓમાંથી મોટી સંખ્યા શોધવા અને તે મોટી સંખ્યા એકી છે કે બેકી તે તપાસવા માટેનો ફ્લોચાર્ટ દોરો.

ઉકેલ :

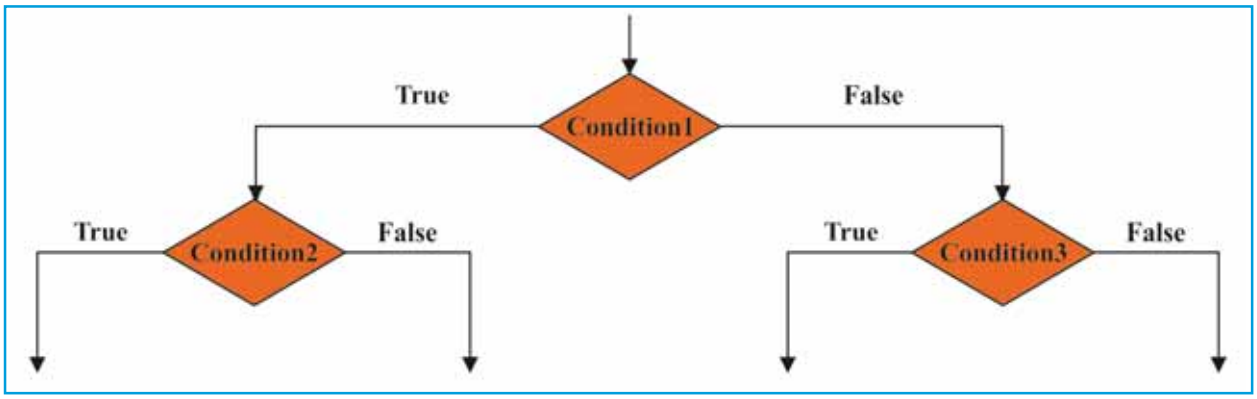
આકૃતિ 5.13માં ફ્લોચાર્ટ દર્શાવેલ છે. જેમ ફ્લોચાર્ટમાં જોઈ શકાય છે, તેમ સૌ પ્રથમ મોટી સંખ્યા શોધવામાં આવે છે (આકૃતિ 5.12 તપાસો) અને તેને Max ચલમાં સંગ્રહિત કરવામાં આવે છે. પછી, નિર્ણય બોક્સનો ઉપયોગ કરીને Max એકી છે કે બેકી તે 2 વડે મોડ્યુલો (% - Modulo) નો ઉપયોગ કરીને ચકાસવામાં આવે છે. (મોડ્યુલો એટલે કે % ઓપરેટર ભાગાકાર પછીની શેષ આપે છે). જો Max%2=0, તો તે બેકી છે, અન્યથા (=1) તે એકી છે. અંતે, આપણે Max ને "Max is Even" અથવા "Max is Odd" લેબલ સાથે પ્રિન્ટ કરીએ છીએ.



આકૃતિ 5.13 : બેમાંથી મોટી સંખ્યા એકી કે બેકી છે તે ચકાસવાનો ફ્લોચાર્ટ

નેસ્ટેડ શરતો અને પુનરાવર્તનના ઉદાહરણો (Nested Conditions and Looping with Examples)

નેસ્ટેડ શરત એટલે શરતની અંદર શરત. ક્યારેક એવું જરૂરી હોય છે કે એક શરત લાગુ કર્યા પછી, આપણે પ્રથમ શરતના પરિણામ (તે સાચું હોય કે ખોટું) ના આધારે બીજી શરત લાગુ કરવાની જરૂર પડે છે. આકૃતિ 5.14માં દર્શાવેલ પરિસ્થિતિને ધ્યાનમાં લો. સૌપ્રથમ, આપણે શરત-1 (Condition1) નું મૂલ્યાંકન કરીએ છીએ. જો તે સાચું હોય, તો આપણે શરત-2 (Condition2) લાગુ કરીએ છીએ અને Condition2 સાચી છે કે ખોટી તેના આધારે આગળ વધીએ છીએ. એ જ રીતે, જો શરત-1 (Condition1) ખોટી હોય, તો આપણે શરત-3 (Condition3) લાગુ કરીશું અને તેના પરિણામ (તે સાચું હોય કે ખોટું) ના આધારે આગળ વધીશું.

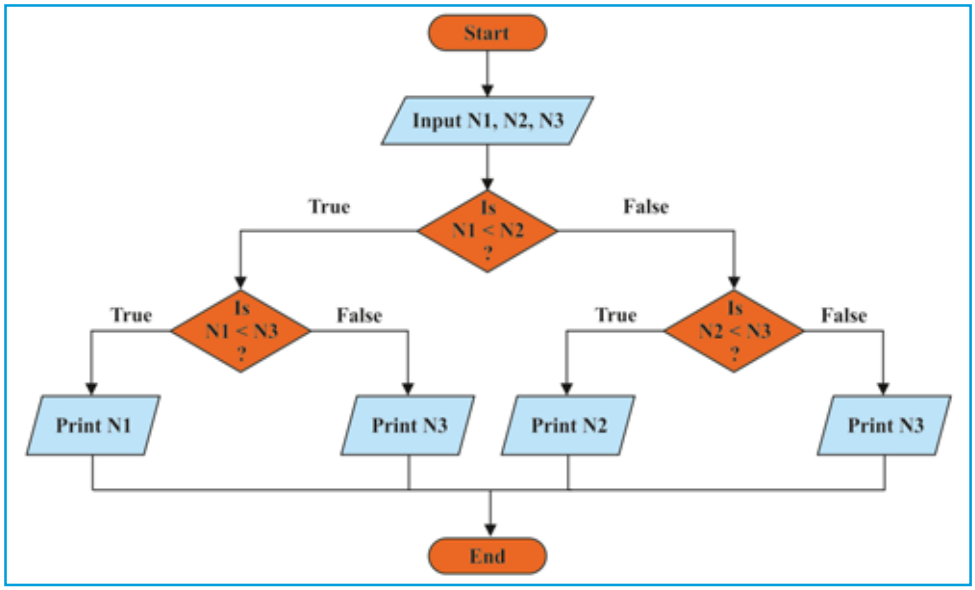


આકૃતિ 5.14 : શરતની અંદર શરત (નેસ્ટેડ શરત)

પ્રશ્ન 6 : ત્રણ સંખ્યાઓમાંથી નાની સંખ્યા શોધવાનો ફ્લોચાર્ટ દોરો.

ઉકેલ :

આકૃતિ 5.15 ત્રણ સંખ્યાઓમાંથી નાની સંખ્યા શોધવા માટેનો ફ્લોચાર્ટ દર્શાવે છે. ફ્લોચાર્ટ સૌપ્રથમ N1, N2 અને N3 ચલનો ઉપયોગ કરીને ત્રણ સંખ્યાઓ સ્વીકારે છે. પછી, નિર્ણય બોક્સ N1 અને N2 ની સરખામણી કરે છે (N1 < N2). જો N1, N2 કરતાં નાનો હોય તો અંદરનો નિર્ણય બોક્સ (ડાબી બાજુ) N1 < N3 નો ઉપયોગ કરીને N1 ની N3 સાથે સરખામણી કરે છે, જેથી N1 નાનો છે કે N3 નાનો છે તે શોધી શકાય. જો N2, N1 કરતાં નાનો હોય તો બીજો અંદરનો નિર્ણય બોક્સ (જમણી બાજુ) N2 < N3 ની ચકાસણી કરે છે, જેથી N2 નાનો છે કે N3 નાનો છે તે શોધી શકાય. અંતે, ત્રણેય સંખ્યાઓમાંથી નાની સંખ્યાને પ્રિન્ટ કરવામાં આવે છે.



આકૃતિ 5.15 : ત્રણ સંખ્યાઓમાંથી નાની સંખ્યા શોધવાનો ફ્લોચાર્ટ

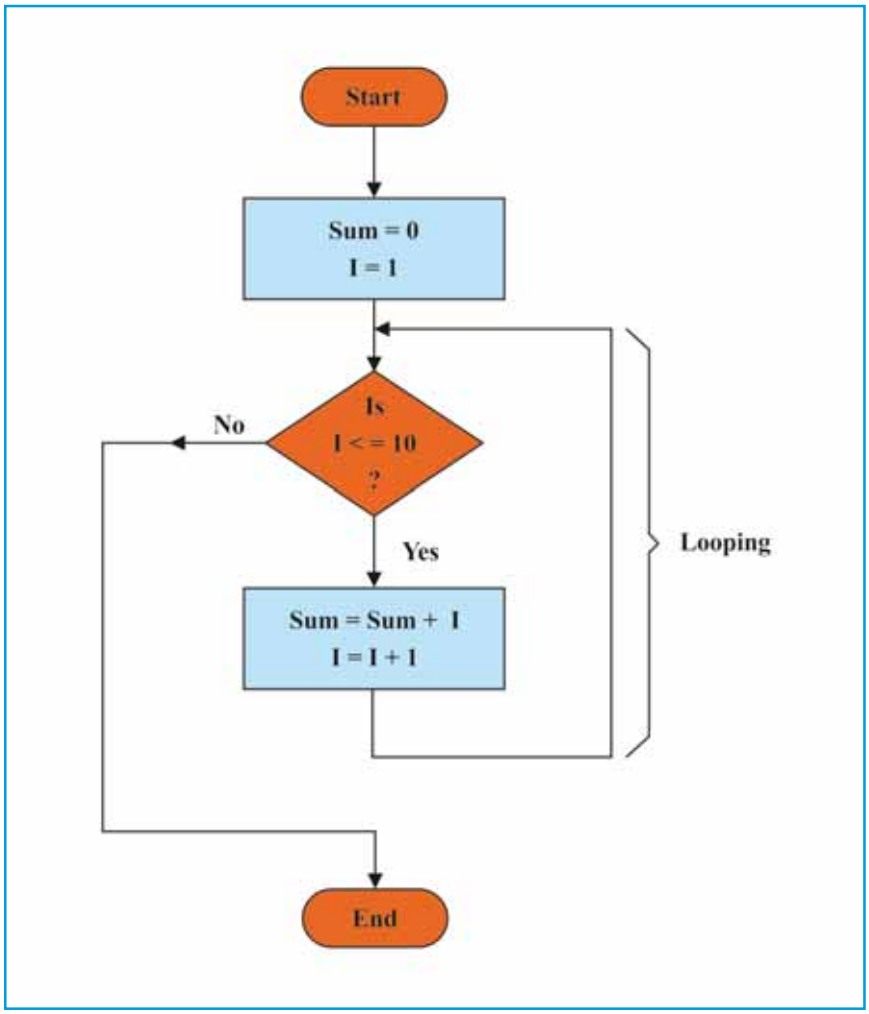
ઉદાહરણ તરીકે, ધારોકે N1 = 5, N2 = 2 અને N3 = 7 છે. શરત N1 < N2 (5 < 2) ખોટી છે. કંટ્રોલ જમણી બાજુ અંદરની શરત તરફ જાય છે જે N2 < N3 (2 < 7) ની ચકાસણી કરે છે, જે સાચી છે. અને અંતે, ત્રણેયમાંથી ન્યૂનતમ સંખ્યા N2 = 2 પ્રિન્ટ થાય છે. તમે N1, N2 અને N3 ના વિવિધ સંયોજનો સાથે આનું પરીક્ષણ કરી શકો છો.

એ પણ શક્ય છે કે કોઈ કાર્યને અમુક શરતને આધારે અમુક પગલાના સમૂહનું પુનરાવર્તન (લૂપિંગ) કરવાની જરૂર હોય. અગાઉ આપણે 1 થી 10 સંખ્યાઓનો સરવાળો કરવાનું ઉદાહરણ લીધું હતું, જેમાં ચલ I નું મૂલ્ય 10 કરતાં ઓછું અથવા તેના બરાબર હોય ત્યાં સુધી દર વખતે ચલ Sum માં આગળનું મૂલ્ય ઉમેરવા અને ચલ I માં 1 નો વધારો કરવાના પગલાનું પુનરાવર્તન કરવું પડતું હતું. ચાલો નીચેના ઉદાહરણ દ્વારા તેને સમજીએ.

પ્રશ્ન 7 : 1 to 10 નો સરવાળો કરવાનો ફ્લોચાર્ટ દોરો.

ઉકેલ :

આકૃતિ 5.16માં ફ્લોચાર્ટ દર્શાવેલ છે. સૌપ્રથમ, પ્રોસેસ બોક્સ Sum = 0 અને I = 1 નું પ્રારંભિક મૂલ્ય સેટ કરે છે. પછી નિર્ણય લેવાની શરત I ≤ 10 લાગુ કરવામાં આવે છે, અને જો તે સાચી હોય, તો Sum = Sum + I અને I = I + 1 ધરાવતું પ્રોસેસ બોક્સનો અમલ કરવામાં આવે છે. ત્યારબાદ એરો ફરીથી નિર્ણય બોક્સ તરફ નિર્દેશ કરે છે અને I ≤ 10 શરત ફરીથી નવી કિંમત સાથે ચકાસવામાં આવે છે. જ્યાં સુધી I ≤ 10 હોય ત્યાં સુધી આ પુનરાવર્તિત થાય છે. જ્યારે I = 11 થાય છે ત્યારે શરત ખોટી થાય છે અને કંટ્રોલ લૂપ પછીના આગળના પગલા પર જાય છે અને સમાપ્ત થાય છે.



આકૃતિ 5.16 : 1 to 10 નો સરવાળો કરવાનો ફ્લોચાર્ટ

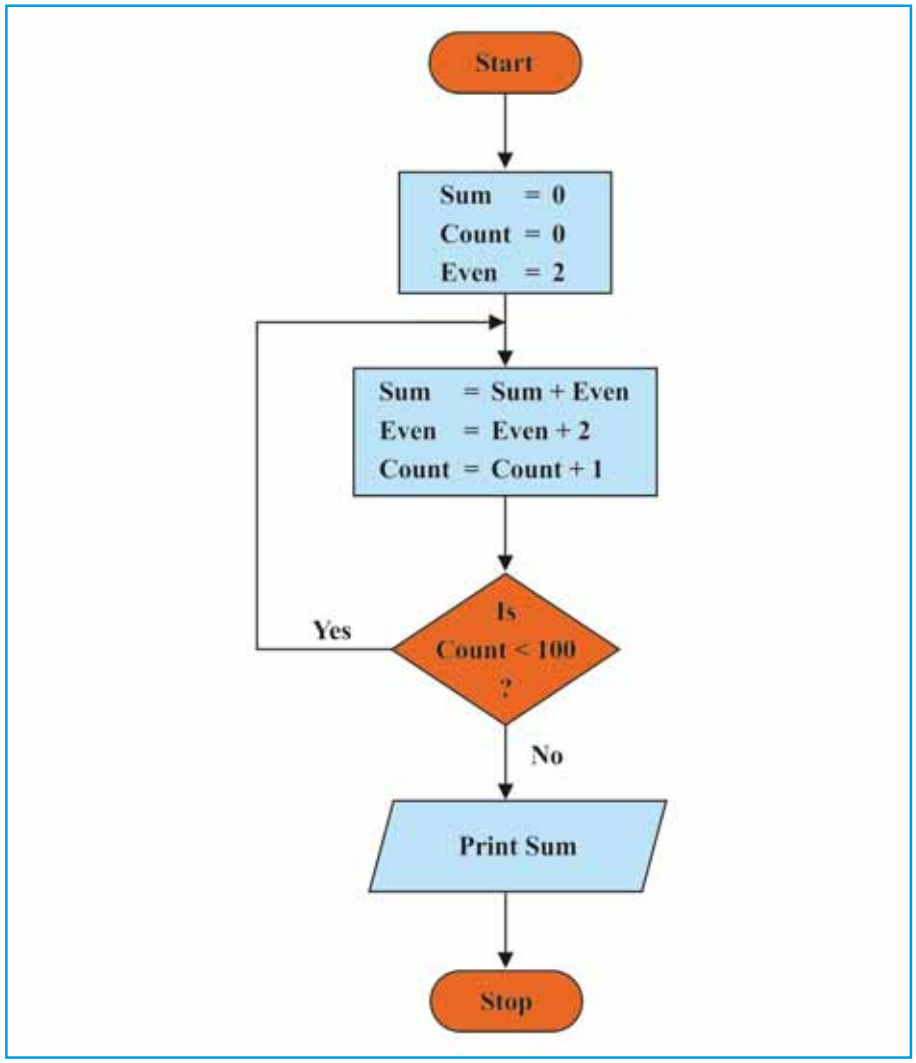
પ્રશ્ન 8 : પ્રથમ 100 બેકી સંખ્યાઓનો સરવાળો કરવાનો ફ્લોચાર્ટ દોરો.

ઉકેલ :

પ્રથમ 100 બેકી સંખ્યાઓ 1 થી 200 ની રેન્જમાં આવરી લેવાય છે અને તેમનો સરવાળો કરવાનો અર્થ છે:

$2 + 4 + 6 + ... + 200$

આકૃતિ 5.17માં ફ્લોચાર્ટ દર્શાવેલ છે. આપણને ત્રણ ચલની જરૂર છે: બધી બેકી સંખ્યાઓનો સરવાળો સંગ્રહવા માટે Sum, કાર્ય ક્યારે સમાપ્ત કરવું તે નક્કી કરવા માટે Count અને આગામી બેકી સંખ્યા સંગ્રહવા માટે Even. તેઓને Sum=0, Count=0 અને Even=2 (પ્રથમ બેકી સંખ્યા) થી પ્રારંભ કરવામાં આવે છે. આપણે Even માં રહેલી બેકી સંખ્યાને ચલ Sum માં ઉમેરીશું, અને પછી Even માં 2 નો ઉમેરો કરીશું અને Count માં 1 નો વધારો કરીશું. આ પ્રક્રિયા કર્યા પછી, ફ્લોચાર્ટ તપાસે છે કે શું Count < 100 છે? જો હા, તો આપણે પ્રક્રિયાનું પુનરાવર્તન કરીશું, અન્યથા કાર્ય સમાપ્ત થાય છે.



આકૃતિ 5.17 : પ્રથમ 100 બેકી સંખ્યાઓનો સરવાળો કરવાનો ફ્લોચાર્ટ

અલ્ગોરિધમનો ટુંકો પરિચય (Overview of Algorithm)

અલ્ગોરિધમ એ આપેલ સમસ્યાનો તાર્કિક ક્રમમાં પગલાં સ્વરૂપે ઉકેલ છે. તે બરાબર એવું જ છે જેમ કોઈ પણ વાનગીની રેસીપી, જે ચોક્કસ ક્રમમાં શું કરવું તે સમજાવે છે. અલ્ગોરિધમ વર્ણનાત્મક શબ્દસમૂહો અથવા સ્યુડો કોડ્સ (pseudo codes)/સૂચનાઓનો ઉપયોગ કરીને લખી શકાય છે. સ્યુડો કોડ્સ ક્રિયાઓ અથવા પગલાં

દર્શાવવા માટે શબ્દો અને પ્રતીકોનો ઉપયોગ કરે છે. તે ખૂબ જ સઘન અને સમજવામાં સરળ હોય છે. અલ્ગોરિધમ સમસ્યાનું નાના પગલાંઓમાં વિઘટન કરે છે, જે સહેલાઈથી સમજી શકાય તેવા હોય છે. અલ્ગોરિધમ કમ્પ્યૂટર પ્રોગ્રામ જેવું જ લાગે છે, સિવાય કે તે કોઈ ચોક્કસ પ્રોગ્રામિંગ ભાષાના ખાસ વાક્યરચનાને બદલે સ્યુડો કોડ્સ જેવી સામાન્ય વાક્યરચનાનો ઉપયોગ કરે છે. અલ્ગોરિધમને કમ્પ્યૂટર પ્રોગ્રામમાં રૂપાંતરિત કરવું ખૂબ જ સરળ છે, કારણ કે સંપૂર્ણ ઉકેલ પહેલેથી જ તાર્કિક ક્રમમાં પગલાંનો ઉપયોગ કરીને વિકસાવવામાં આવેલો હોય છે. પ્રોગ્રામ વિકસાવતા પહેલાં અલ્ગોરિધમ લખવાથી પ્રોગ્રામની સ્પષ્ટતા અને ચોકસાઈ સુનિશ્ચિત થાય છે.

લંબચોરસનું ક્ષેત્રફળ શોધવા માટેના અલ્ગોરિધમનું ઉદાહરણ નીચે મુજબ છે.

1. START
2. Input Length, Breadth
3. Compute Area = Length * Breadth
4. Print Area
5. STOP

નોંધી લો કે અલ્ગોરિધમમાં દરેક પગલાને 1 થી શરૂ કરીને ક્રમ આપવામાં આવે છે. અલ્ગોરિધમની શરૂઆત 'START' થી થાય છે. સ્યુડો કોડ્સ 'Input', 'Compute', 'Print' અનુક્રમે ઇનપુટ, ગણતરી અને આઉટપુટ ક્રિયાઓ દર્શાવે છે. અલ્ગોરિધમનું છેલ્લું પગલું હંમેશા 'STOP' હોય છે, જે અલ્ગોરિધમના અંતને સૂચવે છે. ઉપરોક્ત અલ્ગોરિધમ સૌ પ્રથમ લંબાઈ અને પહોળાઈ મેળવે છે, ત્યારબાદ લંબાઈ અને પહોળાઈનો ગુણાકાર કરીને ક્ષેત્રફળની ગણતરી કરે છે અને અંતે ક્ષેત્રફળને આઉટપુટ તરીકે પ્રિન્ટ કરે છે.

અલ્ગોરીધમના ઉદાહરણો (Examples of Algorithm)

ચાલો આપણે ક્રમિક, નિર્ણય લેવો અને પુનરાવર્તન સહિતના વિવિધ અલ્ગોરિધમ લખીએ, જેથી આપણે તેને ઊંડાણપૂર્વક સમજી શકીએ.

પ્રશ્ન 9 : આપેલ સંખ્યાનો વર્ગ શોધવાનું અલ્ગોરીધમ લખો.

ઉકેલ : અલગોરિધમ નીચે પ્રમાણે છે.

1. START
2. Input N
3. Compute Square = N * N
4. Print Square
5. STOP

અલગોરિધમ ખૂબ જ સરળ છે અને 1 થી 5 સુધીના પગલાંને ક્રમશઃ અનુસરે છે. તે N ની કિંમત સ્વીકારે છે, વર્ગની ગણતરી કરે છે અને પછી વર્ગને પ્રિન્ટ કરે છે.

ચાલો આપણે એવા ઉદાહરણો લઈએ જેમાં નિર્ણય અને પુનરાવર્તન/લૂપિંગનો સમાવેશ થતો હોય.

પ્રશ્ન 10 : બે સંખ્યામાંથી મોટી સંખ્યા શોધવાનું અલગોરિધમ લખો.

ઉકેલ : અલગોરિધમ નીચે પ્રમાણે છે.

1. START
2. Input N1, N2
3. If N1 > N2, goto 6
4. Print N2
5. goto 7
6. Print N1
7. STOP

પ્રથમ અલ્ગોરિધમ, N1 અને N2 ચલનો ઉપયોગ કરીને બે સંખ્યાઓ વાંચે છે. પગલું-3 થી પગલું-6 નિર્ણય લેવાની પ્રક્રિયાનો અમલ કરે છે, જ્યાં પગલું 3 N1 અને N2 ની સરખામણી કરે છે (N1 > N2) અને, જો તે સાચું હોય તો તે N1 ને પ્રિન્ટ કરવા અને ત્યારબાદ 'STOP' કરવા માટે પગલું-6 પર જાય છે. જો તે ખોટું હોય, તો તે N2 ને પ્રિન્ટ કરવા માટે પગલું-4 ને અનુસરે છે અને પગલું-5 નિયંત્રણને 'STOP' કરવા માટે પગલું-7 પર મોકલે છે. જો આપણે પગલું-5 માં 'goto 7' નો ઉપયોગ ન કરીએ, તો N2 પ્રિન્ટ કર્યા પછી તે N1 પણ પ્રિન્ટ કરશે જે ખોટું છે. તેથી, આને ટાળવા માટે કાળજી લેવામાં આવી છે. તેના માટેનો ફ્લોચાર્ટ આકૃતિ 5.12 માં આપેલ છે. અલ્ગોરિધમ અને ફ્લોચાર્ટના પગલાંની તુલના કરો જેથી તે સ્પષ્ટ થશે કે આપણે અલ્ગોરિધમને ફ્લોચાર્ટમાં અને ફ્લોચાર્ટને અલ્ગોરિધમમાં કેવી રીતે રૂપાંતરિત કરી શકીએ છીએ.

પ્રશ્ન 11 : પ્રથમ 100 બેકી સંખ્યાઓનો સરવાળો કરવાનું અલગોરિધમ લખો.

ઉકેલ : અલગોરિધમ નીચે પ્રમાણે છે.

1. START
2. Initialize Sum = 0
3. Initialize Count = 0
4. Initialize Even = 2
5. Compute Sum = Sum + Even
6. Compute Even = Even + 2
7. Increment Count by 1
8. If Count < 100, goto 5
9. Print Sum
10. STOP

પગલાં 2, 3 અને 4 અનુક્રમે Sum, Count અને Even ચલને પ્રારંભિક મૂલ્યો આપે છે. પગલું-5 થી પગલું-8 પુનરાવર્તન (લૂપ) નો અમલ કરે છે. પગલું-5 વર્તમાન Even મૂલ્યને Sum માં ઉમેરે છે. પગલું-6 આગામી બેકી સંખ્યા મેળવવા માટે Even માં 2 ઉમેરે છે. પગલું-7 Count માં 1 નો વધારો કરે છે, કારણ કે એક બેકી સંખ્યા પહેલાથી જ Sum માં ઉમેરવામાં આવી છે. પગલું-8 તપાસે છે કે Count 100 કરતાં ઓછો છે કે કેમ. જો તે સાચું હોય, તો તે પરત પગલું-5 પર જાય છે અને લૂપનું પુનરાવર્તન ફરી થાય છે. જ્યારે Count = 100 થાય છે, ત્યારે શરત ખોટી પડે છે અને તે Sum ને પ્રિન્ટ કરવા માટે પગલું-9 ને અનુસરે છે. આ રીતે, પગલું-5 થી પગલું-8, 100 વખત પુનરાવર્તિત થાય છે અને દરેક વખતે આગામી બેકી સંખ્યા (2, 4, 6, ..., 200) Sum માં ઉમેરવામાં આવે છે અને Count 0 થી 99 સુધી જાય છે અને જ્યારે તે 100 થાય છે, ત્યારે લૂપ સમાપ્ત થાય છે. તેના માટેનો ફ્લોચાર્ટ આકૃતિ 5.17 માં આપેલ છે. લૂપનો અમલ કેવી રીતે થાય છે તે સમજવા માટે તેની તુલના ઉપરોક્ત અલ્ગોરિધમ સાથે કરો.

ફ્લોચાર્ટ વિરુદ્ધ અલ્ગોરિધમ (Flowchart Vs. Algorithm)

આપણે ઉદાહરણો સાથે ફ્લોચાર્ટ અને અલ્ગોરિધમનો અભ્યાસ કર્યો અને સમજ્યા કે ક્રમ, નિર્ણય અને લૂપિંગનો અમલ કેવી રીતે કરી શકાય, જે સમસ્યાના ઉકેલ માટેના સૌથી મહત્ત્વપૂર્ણ ઘટકો છે. વાસ્તવિક જીવનની કોઈપણ સમસ્યા આ ત્રણેય પ્રકારનું અલગ-અલગ સંયોજન હોય છે. જરૂરિયાત અથવા પસંદગીના આધારે, આપણે પ્રક્રિયાને - એટલે કે સમસ્યાના ઉકેલને - સંરચિત અને તાર્કિક ક્રમમાં પગલાં સ્વરૂપે રજૂ કરવા માટે ફ્લોચાર્ટ અથવા અલ્ગોરિધમ અથવા બંનેનો ઉપયોગ કરી શકીએ છીએ. જોકે ફ્લોચાર્ટ અને અલ્ગોરિધમ બંનેનો ઉપયોગ સમસ્યાના ઉકેલની પદ્ધતિઓ તરીકે થાય છે, તેમ છતાં તેમની પોતાની યોગ્યતાઓ અને ખામીઓ છે. ચાલો આપણે ફ્લોચાર્ટ અને અલ્ગોરિધમની તુલના કરીએ. તે નીચેના ટેબલમાં આપેલ છે.

ફ્લોચાર્ટ	અલ્ગોરિધમ
પ્રતીકોનો ઉપયોગ કરીને પ્રક્રિયાનું દ્રશ્ય નિરૂપણ (visual representation) પૂરું પાડે છે.	પ્રક્રિયાનું નિરૂપણ કરવા માટે સ્યુડો કોડ્સ (pseudo codes) અને સૂચનાઓનો ઉપયોગ કરે છે.
બધા પાથ દશ્યમાન હોય છે અને અનુસરવા માટે સરળ હોય છે.	માર્ગો છુપાયેલા હોય છે અને સમજવા માટે પગલાં વાંચવાની જરૂર પડે છે.
પ્રક્રિયાનો પ્રવાહ દર્શાવે છે, જે દ્રશ્યથી શીખનારાઓ માટે સારું છે.	પગલાંનો ઉપયોગ કરીને પ્રક્રિયાનો તર્ક પૂરો પાડે છે, જે તાર્કિક વિચારકો માટે સારું છે.
પ્રક્રિયાનું આયોજન અને સમજાવવા માટે સારું છે.	કમ્પ્યૂટર પ્રોગ્રામ લખવા માટે સારું છે, કારણ કે અલ્ગોરિધમ કમ્પ્યૂટર પ્રોગ્રામ જેવું જ હોય છે.
નવા લોકો માટે સારું છે.	કમ્પ્યૂટર પ્રોગ્રામર માટે સારું છે.

સારાંશ

આ પ્રકરણની શરૂઆત આપણે સમસ્યાના ઉકેલ અને તેના મહત્વ વિષે રજૂઆત સાથે કરી. સમસ્યાનો ઉકેલ એ કોઈપણ સમસ્યાનો ઉકેલ શોધવા માટેની એક તાર્કિક અને ક્રમબદ્ધ પ્રક્રિયા છે. સમસ્યાનું નિરાકરણ કમ્પ્યૂટર સાયન્સના કેન્દ્રમાં છે અને તે કમ્પ્યૂટર સાયન્સ તેમજ દૈનિક જીવન માટે એક મૂળભૂત કૌશલ્ય છે. આપણે સમસ્યાઓ ઉકેલવા માટેના અભિગમનો ઉપયોગ કરતી બે બહોળા પ્રમાણમાં વપરાતી પદ્ધતિઓ ફ્લોચાર્ટ અને અલ્ગોરિધમનો અભ્યાસ કર્યો છે. ફ્લોચાર્ટએ ઉકેલનો પ્રવાહ દર્શાવવા માટે પ્રતીકોનો ઉપયોગ કરીને ઉકેલનું ચિત્રાત્મક નિરૂપણ છે. વધુ સારી સમજણ માટે, આપણે ઘણી નાની અને સરળ સમસ્યાઓ માટે ફ્લોચાર્ટ્સ વિકસાવ્યા છે. ઉદાહરણો દ્વારા આપણને નિર્ણયના આધારે ફ્લોચાર્ટના વિવિધ માર્ગો અનુસરી સમજવાની તકો પણ મળી. અલ્ગોરિધમ સમસ્યાનો પગલાં સ્વરૂપે ઉકેલ પૂરો પાડે છે. વધુ સારી સમજણ અને સ્પષ્ટતા માટે, આ પ્રકરણમાં જે સમસ્યા માટે ફ્લોચાર્ટ દોર્યા, તે જ સમસ્યાઓ માટે અલ્ગોરિધમ લખવાના ઉદાહરણો આપણે લીધાં. આ પ્રકરણનો અંત આ બંને ઉકેલ પદ્ધતિઓ, એટલે કે ફ્લોચાર્ટ અને અલ્ગોરિધમ, વચ્ચેની તુલના સાથે થાય છે. આ પદ્ધતિઓ તાર્કિક વિચારસરણી અને ઉકેલો વ્યક્ત કરવામાં સ્પષ્ટતા લાવે છે. તે વિદ્યાર્થીઓને વિવિધ ઇનપુટ્સ આપી, નિર્ણયોના આધારે વિવિધ માર્ગોને અનુસરીને ઉકેલનું પરીક્ષણ કરવાની પણ મંજૂરી આપે છે. આ પ્રકરણે આવનારા પ્રકરણોમાં C પ્રોગ્રામિંગ શીખવા માટે એક મજબૂત પાયો નાખ્યો છે.

સ્વાધ્યાય

1. સમસ્યા નિરાકરણ એટલે શું? એક ઉદાહરણ આપો.
2. સમસ્યા નિરાકરણની પદ્ધતિઓ અને તેના ફાયદાઓ જણાવો.
3. ફ્લોચાર્ટ શું છે? ફ્લોચાર્ટમાં વપરાતા પ્રતીકો અને તેમનું કાર્ય દર્શાવો.
4. ફ્લોચાર્ટના પ્રોસેસ બોક્સમાં સામાન્ય રીતે કયા પ્રકારના કાર્યોનો સમાવેશ થાય છે?
5. નિર્ણય લેવા માટે કયા પ્રતીકનો ઉપયોગ થાય છે? ઉદાહરણ સાથે તેનો ઉપયોગ સમજાવો.
6. ફ્લોચાર્ટ પ્રતીકોનો ઉપયોગ કરીને તમે પુનરાવર્તન કેવી રીતે કરશો?
7. અલ્ગોરિધમ શું છે? એક સરળ ઉદાહરણ આપો.
8. અલ્ગોરિધમનો ઉપયોગ કરીને પુનરાવર્તનનું એક ઉદાહરણ આપો.
9. અલ્ગોરિધમના ગેરફાયદાઓ જણાવો.
10. ફ્લોચાર્ટ અને અલ્ગોરિધમની સરખામણી કરો.

11. સાચું કે ખોટું જણાવો.

(1) ફ્લોચાર્ટમાં બે પગલાંને જોડવા માટે એરોનો ઉપયોગ થાય છે.

(2) અલ્ગોરિધમમાં તમામ સંભવિત માર્ગો દૃશ્યમાન હોય છે.

(3) સમસ્યા નિરાકરણ એ સમસ્યાને ઉકેલવા માટેની તાર્કિક અને ક્રમબદ્ધ પ્રક્રિયા છે.

(4) બહુ-પસંદગી ડાયમંડ દ્વારા અમલમાં મૂકી શકાય છે.

(5) ફ્લોચાર્ટમાં હંમેશા Start અને End (શરૂઆત અને અંત) પગલાં હોય છે.

12. ખાલી જગ્યાઓ પૂરો.

(1) _____ ઉકેલનું ચિત્રાત્મક નિરૂપણ છે.

(2) અલ્ગોરિધમમાં _____ કોડનો ઉપયોગ થાય છે.

(3) ફ્લોચાર્ટમાં કનેક્ટર્સ (Connectors) _____ પ્રતીકનો ઉપયોગ કરે છે.

(4) સામાન્ય રીતે અંકગણિત અને તાર્કિક કામગીરીઓ _____ બોક્સમાં મૂકવામાં આવે છે.

(5) સમસ્યા નિરાકરણ માટેની _____ અને _____ પદ્ધતિઓ છે.

13. બહુવિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.

(1) ફ્લોચાર્ટની શરૂઆત અને અંત માટે નીચેનામાંથી કયું પ્રતીક વપરાય છે?

(a) લંબચોરસ (b) ડાયમંડ (c) અંડાકાર (d) વર્તુળ

(2) નીચેનામાંથી કયું સમસ્યાના ઉકેલનું દ્રશ્ય નિરૂપણ પૂરું પાડે છે?

(a) અલ્ગોરિધમ (b) ફ્લોચાર્ટ (c) પ્રોગ્રામ (d) પક્રિયા

(3) નીચેનામાંથી કયું પ્રતીક ફ્લોચાર્ટમાં પ્રક્રિયા દર્શાવવા માટે વપરાય છે?

(a) લંબચોરસ (b) ડાયમંડ (c) અંડાકાર (d) વર્તુળ

(4) નીચેનામાંથી કયું કનેક્ટર જોડીઓને એકબીજાથી અલગ પાડવા માટે વપરાય છે?

(a) એરોની દિશા (b) વર્તુળમાં લખેલ અક્ષર

(c) વર્તુળનું કદ (d) વર્તુળનો રંગ

(5) સરખામણી નીચેમાંથી શેમાં વપરાય છે?

(a) પ્રક્રિયા (b) નિર્ણય-કરતાં (c) ઈનપુટ (d) આઉટપુટ

(6) નીચેનામાંથી કયું ઉકેલની અંદરના જુદા જુદા પાથ ઓળખવા માટે શ્રેષ્ઠ છે?

(a) ફ્લોચાર્ટ (b) અલ્ગોરિધમ

(c) ફ્લોચાર્ટ અને અલગોરિધમ બંને (d) આપણે પાથ જોઈ ન શકીએ

(7) નીચેનામાંથી કયું ઉકેલના પગલાં દર્શાવવા માટે સ્યુડો કોડ્સનો ઉપયોગ કરે છે?

(a) પ્રોગ્રામ (b) ફ્લોચાર્ટ (c) અલ્ગોરિધમ (d) પ્રક્રિયા

(8) નીચેનામાંથી કયું સામાન્ય રીતે ગણતરીઓ દર્શાવવા માટે વપરાય છે?

(a) પ્રોસેસ બોક્ષ (b) નિર્ણય બોક્ષ (c) ઈનપુટ બોક્ષ (d) આઉટપુટ બોક્ષ

(9) નીચેનામાંથી કયું ફ્લોચાર્ટને એક કરતાં વધુ પૃષ્ઠોમાં વિભાજિત કરવા માટે વપરાય છે?

(a) એરો સાથે અક્ષર (b) કનેક્ટર્સ (c) એરો (d) ખાસ પ્રતીકો

(10) જ્યારે આપણે બે કિંમતો ગણવાની હોય ત્યારે તેને

(a) એક જ બોક્ષમાં એક પછી એક મૂકો

(b) તેઓને બે જુદા બોક્ષમાં એક પછી એક મૂકો

(c) a અને b બંને

(d) બે જુદા બોક્ષમાં એક પછી એક એમ જ હોવા જોઈએ

પ્રાયોગિક સ્વાધ્યાય

1. C = (5.0/9.0) × (F–32) સૂત્રનો ઉપયોગ કરીને ફેરનહીટને સેલ્સિયસમાં રૂપાંતરિત કરવા માટેનો ફ્લોચાર્ટ દોરો.
2. આપેલી ધન સંખ્યા એકી છે કે બેકી તે ચકાસવા માટેનો ફ્લોચાર્ટ દોરો.
3. 1 થી 100 સુધીની માત્ર એકી સંખ્યાઓનો સરવાળો કરવા માટેનો ફ્લોચાર્ટ દોરો.
4. વિદ્યાર્થીના 100 માંથી ગુણ સ્વીકારવા અને જો ગુણ ≤ 35 હોય તો "Fail", જો ગુણ > 35 અને ≤ 60 હોય તો "Second class", જો ગુણ > 60 અને ≤ 70 હોય તો "First class", અન્યથા "Distinction" પ્રિન્ટ કરવા માટેનો ફ્લોચાર્ટ દોરો.
5. એક સંખ્યાનું અનુમાન કરવા માટેનો ફ્લોચાર્ટ દોરો. તે વપરાશકર્તા પાસેથી સંખ્યા 7 ન હોય ત્યાં સુધી વારંવાર સંખ્યા સ્વીકારે છે. જ્યારે સંખ્યા 7 હોય, ત્યારે તે "Congraulations!" પ્રિન્ટ કરે છે અને સમાપ્ત થાય છે.
6. આપેલી 10 સંખ્યાઓમાંથી લઘુત્તમ સંખ્યા શોધવા માટેનો ફ્લોચાર્ટ દોરો.
7. આપેલી બે સંખ્યાઓના સરવાળા અને તફાવત શોધવા માટેનું અલ્ગોરિધમ લખો.
8. ત્રણ સંખ્યાઓમાંથી મહત્તમ સંખ્યા શોધવા માટેનું અલ્ગોરિધમ લખો.
9. ઇનપુટ તરીકે આપેલી N સંખ્યાઓમાંથી માત્ર બેકી સંખ્યાઓનો સરવાળો કરવા માટેનું અલ્ગોરિધમ લખો.
10. આપેલી શ્રેણીનો સરવાળો કરવા માટેનો અલ્ગોરિધમ લખો : 1 – 2 + 3 – 4 + 5 – 6 + ... + N.