1

प्रोग्रामिंग मेथेडोलोजी
(Programming Methodlogy)

1.1 एल्गोरिथम (Algorithm)

किसी प्रोग्राम के हल को अलग–अलग स्टेप के अंदर दिखाने को एल्गोरिथम कहते है।

जब हम किसी एल्गोरिथम का प्रयोग करेंगे तो यह सुनिश्चित करेंगे कि हर ऑपरेशन निश्चित समय तक निष्पादित हो। यानि जितनी बार भी स्टेप निष्पादित (Execute) हो वह एक निश्चित संख्या तक हो। तथा हर स्टेप प्रभावकारी (Effective) हो।

1.2 एल्गोरिथम का बनाना (Develop an Algorithm)

एल्गोरिथम बनाने के लिए जो भी प्रोग्राम दिया गया है उसका ढंग से अध्ययन करना चाहिये। प्रोग्रामर को जो भी प्रोग्राम बनाना है। उसको ढंग से समझकर फिर उस प्रोग्राम का तरीका व तर्क निकाले जिससे कि उस प्रोग्राम का हल निकाल सके। तर्क हमेशा सीधा और समझने वाला होना चाहिये।

साधारणतया एल्गोरिथम बनाते समय निम्न बातों का ध्यान रखना चाहिये।

1. सबसे पहले दिये हुए प्रश्न को समझें।
2. फिर प्रश्न का उत्तर (Output) समझने का प्रयत्न करना चाहिये।
3. फिर प्रश्न के लिए जो भी इनपुट (Input) की जरूरत है। जिससे कि उत्तर सही ढंग से आ जाये, पता करना चाहिये।
4. फिर जो भी तर्क और हल निकाला है। उसका नक्शा बनाना चाहिये जिससे वह प्रश्न का सही उत्तर आ जाये।
5. एल्गोरिथम को इनपुट डाटा देकर जांच करनी चाहिये।
6. स्टेप 1 से 5 तब तक दोहराये जब तक कि एल्गोरिथम द्वारा सही उत्तर नहीं आ जाये।

जहां भी हम एल्गोरिथम में लिखे $x = x + 5$ तो इसका मतलब है कि x का नया मान = x का पुराना मान + 5 अर्थात x का पुराना मान इस कथन को निष्पादित करने के बाद खत्म हो गया है। और उसकी जगह x के नये मान ने ले ली है।

उदाहरण :– किसी भी नंबर का फेक्टोरियल निकालने का एल्गोथ्रिम लिखो।

```
1    Read n
2    F ⟵ 1
3    I ⟵ 1
4    5 से 6 तक लाइन को दोहराये जब तक I<=n हो
5    F ⟵ F*I
6    I ⟵ I+1
7    Print F
```

1. एक एल्गोरिथम बनाइये जो कि प्रथम n प्राकृत संख्याओं के वर्ग का जोड़ निकाले।

```
Step 1.    Read n
Step 2.    Sum ⟵ 0
Step 3.    Set i ⟵ 1
Step 4.    if i<=n
Step 5.    Sum ⟵ Sum+i*i
Step 6.    Set i ⟵ i+1
Step 7.    go to step 4
Step 8.    Write sum
```

उदाहरण 2. एक एल्गोरिथम बनाइये जो कि प्रथम n संख्याओं के सम तथ विषम संख्याओं का जोड़ निकाले।

```
Step 1.    Read n
Step 2.    Se ⟵ 0
Step 3.    So ⟵ 0, i ⟵ 1
Step 4.    if i<=n
Step 5.    if (i%2=0)      Se ⟵ Se+i
Step 6.    else so ⟵ so+i
Step 7.    set i ⟵ i+1
Step 8.    Go to step 4
Step 9.    Print Se, So
```

1.3 मॉड्यूलर प्रस्ताव (Modular Approach)

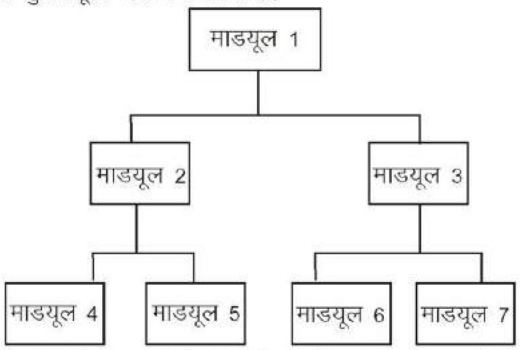
मॉड्यूलर का मतलब होता है छोटा प्रोग्राम। मॉड्यूलर प्रस्ताव एक प्रकार की तकनीक है। जिसमें यदि कोई प्रोग्राम कठिन लगता है तो, उसको अलग–अलग दो प्रोग्राम में बाँट देते है। अब इन अलग–अलग प्रोग्रामों को अलग–अलग कम्प्यूटर में रन करते हैं। और इनकी कमियों (Error) को दूर करते हैं।

इस प्रकार इस तकनीक को स्ट्रक्चर्ड प्रोग्रामिंग (Structured Programming) भी कहते है। क्योंकि इसमें हर कठिन प्रोग्राम को अलग–अलग बाँट देते हैं। अलग–अलग प्रोग्राम में जाने (Entry)

का और बाहर निकालने (Exit) का केवल एक ही रास्ता होता है।

(i) ऊपर से नीचे प्रस्ताव (Top down approach)

इस प्रस्ताव में बड़े प्रोग्राम को छोटे–छोटे प्रोग्राम में बांट देंगे। प्रोग्राम सबसे पहले मुख्य प्रोग्राम से प्रारंभ होगा। जो कि नीचे आते आते छोटे छोटे प्रोग्रामों में बंटता जायेगा। अब हर छोटे प्रोग्रामों को हम अलग–अलग से सुविधापूर्वक हल कर सकते है।



साधारणतया जब भी कोई प्रोग्राम बनाते है तो इसी तकनीक का प्रयोग करते है।

उदाहरण के लिए यदि हमें कोई बड़े मकान का निर्माण करते है तो हम अपनी सुविधानुसार पहले सभी कमरों का निर्माण कार्य करायेंगे जो कि ईट व सीमेन्ट से होगा, उसके बाद हम सारा लोहे का काम करायेंगे। उसके बाद लकड़ी का काम करायेंगे। कहने का मतलब है कि हम कभी भी यह नहीं करायेंगे कि हम एक कमरा बनायेंगे और उसमें पहले चुनाई का काम, फिर लकड़ी का काम, फिर लोहे का काम करायेंगे। और उसके बाद फिर दूसरे कमरे का काम भी इसी तरह करायेंगे।

1.4 प्रलेखन (Documentation)

प्रलेखन किसी भी समस्या के डाटा का समूह व उसके हल को सुव्यवस्थित तरीके से कागज पर दर्शाने को कहते है। प्रलेखन के अंतर्गत यह भी बताना पड़ता है कि यह समस्या (problem) किसने लिखी, कब लिखी और क्यों लिखी। इस समस्या के अंदर कौन–कौनसे फंक्शन का प्रयोग किया गया है।

किसी भी सॉफ्टवेयर के प्रलेखन में निम्नलिखित चीजें जरूर होनी चाहिये।

1.4.1 सॉफ्टवेयर को परिभाषित करना :– प्रलेखन को परिभाषित करते समय हम यह जरूर बताये कि सॉफ्टवेयर किस चीज का है। तथा किसके द्वारा बनाया गया है। और किस काम के लिए बनाया गया है।

1.4.2 सॉफ्टवेयर को वर्णन करना (Description of the program) :– सॉफ्टवेयर का वर्णन करने के लिए फ्लोचार्ट, एल्गोरिथम, जॉच के लिए डाटा जो प्रयोग में लिए, जॉच के बाद जो परिणाम आये आदि सब चीजों का प्रलेखन के अन्तर्गत समावेश होना चाहिये।

1.4.3 निकाय के बारे में वर्णन (Description of the system)

प्रलेखन के अन्तर्गत हमको यह भी बताना है कि यह सॉफ्टवेयर किस निकाय या उप निकाय वातावरण में अच्छी तरह काम करता है। उस निकाय का पूरी तरह वर्णन करना है। अगर निकाय ज्यादा

बड़ा है तो संक्षिप्त लाइनों में उस निकाय का वर्णन करना है। किस तरह के इनपुट डाटा प्रोग्राम के अंदर प्रयोग में लिये गये है। तथा किस तरह का आउटपुट प्रोग्राम द्वारा मिलेगा आदि सब कथनों का स्पष्ट वर्णन होना चाहिये।

1.4.4 ऑपरेटर निर्देश पुस्तिका (An operator instruction manual)

सॉफ्टवेयर पैकेज के अंदर ऑपरेटर निर्देश पुस्तिका होनी चाहिए। ऑपरेटर निर्देश पुस्तिका के अंतर्गत हर निर्देश को लिखना कि इस सॉफ्टवेयर को चलाने के लिए कम्प्यूटर को क्या–क्या निर्देश देने होंगे। प्रत्येक निर्देश स्पष्ट होने चाहिये जिससे एक साधारण व्यक्ति भी उन निर्देशों का सरलता से प्रयोग कर सके। निर्देश पुस्तिका में यह निर्देश भी स्पष्ट होना चाहिये कि इस सॉफ्टवेयर का कम्प्यूटर में कैसे प्रयोग करेंगे तथा इस सॉफ्टवेयर से कैसे बाहर निकलेंगे।

1.5 प्रोग्राम का रखरखाव (Maintaining the program)

एक बार प्रोग्राम लिखने के बाद उसमें समय की आवश्यकतानुसार परिवर्तन करने पड़ते है। इस आवश्यकतानुसार परिवर्तन को ही प्रोग्राम का रखरखाव कहते है। रखरखाव कई बार तो प्रोग्राम का प्रयोग करने वाले की आवश्यकता के अनुसार किया जाता है। और कई बार नियम परिवर्तन के कारण करना पड़ता है।

उदाहरण के लिए एक प्रोग्राम है जो कि प्रोग्रामर A ने राजस्थान सरकार के कर्मचारियों के वेतन की गणना करने के लिए बनाया है। जब यह प्रोग्राम या सॉफ्टवेयर लिखा था, उस समय यह राजस्थान सरकार द्वारा निर्देशित सभी शर्तो की पालना करता था। जैसे ही पाँचवा वेतन आयोग लागू हुआ। पुराना प्रोग्राम तो पूरी तरह से काम करेगा साथ ही अब इस प्रोग्राम को नये नियमों के तहत आधुनिक बनाने के लिए इसके अंदर परिवर्तन करना होगा। और उस परिवर्तन के अंदर जो नये नियम पाँचवे वेतन आयोग में है उनका समावेश करना पड़ेगा।

इस तरह से समय–समय पर एक प्रोग्राम का रखरखाव करना बहुत जरूरी है।

1.6 डिबगिंग प्रोग्राम (Debugging Program) बग (Bug) का मतलब होता है त्रुटि (error) और डि+बग यानि डि का मतलब हटाना, बग (Bug) का मतलब त्रुटि अर्थात त्रुटि को हटाना।

डिबगिंग एक तरीका है जिसमें हम प्रोग्राम के अंदर त्रुटि का पता करते है। और पता करने के बाद उस त्रुटि को सही करते है।

किसी भी प्रोग्राम के अंदर त्रुटि को हटाने के निम्न तरीके अपनाये जाते है।

1. किसी प्रोग्राम के छोटे भाग को, मुख्य प्रोग्राम में जोडने से पहले उसे अलग से कम्प्यूटर पर रन करके देख ले कि इस छोटे प्रोग्राम के अंदर कोई त्रुटि है या नहीं है। त्रुटि की जांच करने के पश्चात् ही उस छोटे प्रोग्राम को, मुख्य प्रोग्राम में जोड़ा जाये।
2. इसके अन्तर्गत हम प्रोग्राम के अंदर अतिरिक्त वेरियबल लेकर उनकी आउटपुट की जांच प्रोग्राम के अंदर बीच–बीच में करते रहते है। जिससे कि हम त्रुटि का पता लगा सकते है।
3. डाटा को ट्रेस (Trace) करना।

1.7 सिन्टेक्स त्रुटि (Syntax error)

सिन्टेक्स त्रुटि तब आती है जब किसी भाषा की ग्रामर का नियम पालना सही ढंग से नहीं किया गया हो। यह बहुत सामान्य त्रुटि है जो कि अधिकतर आती रहती है इस त्रुटि को ठीक करना भी सरल

है। जब भी इस तरह की त्रुटि प्रोग्राम में आती है तो कम्पाइलर उसी समय बता देता है। जिससे कि त्रुटि को दूर करना सुविधाजनक रहता है। तथा यह भी पता लग जाता है कि कहां पर त्रुटि है। और क्या है।

उदाहरण के लिए–

```
# include <stdio.h>
void main ( )
{
printf ("\n enter the value of n");
scanf ("%", & n)
}
```

उपरोक्त प्रोग्राम को जब हम कम्प्यूटर पर (Compile) कम्पाइल करते है। तब वह दो सूचना देगा। एक यह कि जो वेरियबल n का प्रयोग किया है। वह प्रोग्राम में डिक्लयेर नहीं किया गया है। दूसरी यह कि scanf के बाद सेमीकॉलान (;) का प्रयोग नहीं किया गया है।

1.8 रन टाइम त्रुटि (Run time error)

जब हम किसी प्रोग्राम को कम्पाइल करते है तो वह किसी भी प्रकार की त्रुटि नहीं बताता है। लेकिन जब उस प्रोग्राम को कम्प्यूटर पर रन करते है, तो त्रुटि आती है। इस प्रकार की त्रुटि को रन टाइम त्रुटि कहते है।

उदाहरण के लिए हम संख्या में भाग देने का प्रोग्राम बनाते है।

```
# include <stdio.h>
# include <conio.h>
void main ( )
{
int a, b, c
printf ("\n enter value of a");
scanf ("%d", &a);
printf("\n enter value of b");
Scanf("%d," & b);
c=a/b;
printf("\n division=%d," c);
}
```

उपरोक्त प्रोग्राम को जब भी इस हम कम्प्यूटर पर कम्पाइल करेंगे तो यह किसी भी प्रकार की त्रुटि नहीं दर्शायेगा। लेकिन इसी प्रोग्राम को जब हम कम्प्यूटर पर रन करेंगे और a का मान 6 व b का मान 0 देंगे तो यह त्रुटि दर्शायेगा। इस प्रकार की त्रुटि को रन टाइम त्रुटि कहते है। क्योंकि किसी भी नंबर को अगर हम 0 से भाग देंगे तो परिणाम संभव नहीं है। क्योंकि कोई भी संख्या को 0 से भाग देंगे तो परिणाम अनन्त (∞) आता है।

इसी तरह अगर हम किसी संख्या का (log) लॉग निकालने का प्रोग्राम बनाये तो वह भी जब कम्पाइलर के द्वारा आब्जेक्ट प्रोग्राम (object program) में परिवर्तित करेगा, तब वह किसी भी प्रकार की त्रुटि नहीं बतायेगा। लेकिन जब हम इसी प्रोग्राम को कम्प्यूटर पर रन करे और ऋणात्मक संख्या दे तो

हमको प्रोग्राम में त्रुटि मिलेगी। जो कि रन टाइम त्रुटि है।

1.9 तर्क त्रुटि (Logical Error)

सिन्टेक्स त्रुटि और रन टाइम त्रुटि के बाद तर्क त्रुटि (Logical Error) आती है। यह त्रुटि सबसे ज्यादा कठिन व सबसे ज्यादा समय लगाने वाली होती है। इस त्रुटि को पता करने में सबसे ज्यादा समय लगता है।

तर्क त्रुटि साधारणतया जब आती है, जब किसी प्रोग्राम के कथन को गलत तरीके से लिख देते है।

सबसे अच्छा उदाहरण है कि जब किसी भी द्विघात समीकरण $ax^2+bx+c=0$ के मूल निकालने के लिए हल करते है तो हम निम्न प्रकार लिखते है।

```
d = b*b - 4*a*c;
Root 1 = (-b+Sqrt (d))/ 2*a
```

उपरोक्त प्रोग्राम में अगर हम यह कोड लिखकर कम्पाइल करेंगे तो, कम्पाइलर किसी भी प्रकार की त्रुटि नहीं बतायेगा। और जब इसी प्रोग्राम को रन करेंगे और a, b और c का मान देंगे तो हम कभी भी इस समीकरण के मूल सही नहीं निकाल पायेंगे। कारण यह है कि फार्मूला के हिसाब से पहले यह ब्रेकेट के अंदर का मान ज्ञात करेंगे। उसके बाद इस मान को 2 से भाग देंगे। उसके बाद जो भी परिणाम आयेगा उसको a से गुणा करेंगे। जबकि हमारा वास्तविक सूत्र पहले ब्रेकेट के अंदर का मान निकालकर उसको 2a से भाग देना है। तो इस उपरोक्त प्रोग्राम से सही मूल निकालने के लिए 2*a को ब्रेकेट के अंदर लिखेंगे।

```
Root 1 = (-b+Sqrt (d)) / (2*a);
```

1.10 प्रोब्लम सोल्विंग मेथोडोलोजी (Problem solving methodology)

अब सारे नियम जानने के बाद अगला कार्य है। इन नियमों की पालना करना और प्रश्न को हल करना।

1.10.1 समस्या को समझना (Understanding the problem)

इसके अन्तर्गत हमको जो भी समस्या दी हुई है उसको समझना है।

उदाहरण के लिए यदि एक प्रोग्राम फैक्टोरियल निकालने का या प्राईम नम्बर का पता करने का है। इस समस्या को हल करने के पहले फैक्टोरियल को गणना करने का सूत्र तथा प्राईम (Prime) नम्बर का पता करने का तरीका मालूम होना चाहिए। अन्यथा इस समस्या को हल नहीं कर सकते।

1.10.2 कम से कम इनपुट का पता करना (Identifying Minimum Number of Inputs Required)

समस्या को समझने के बाद अगला कार्य समस्या से संबंधित इनपुट का पता करना है कि इस समस्या को हल करने के लिए कितने इनपुट वेरियवल की आवश्यकता है। जिससे कि जो समस्या दी हुई है वह हल हो जाये और उसका आउटपुट मिल जाये।

उदाहरण के लिए हमको ब्याज निकालने के लिए प्रोग्राम बनाना है तो हम इस प्रोग्राम को हल करने के लिए तीन इनपुट वेरियवल P,R,T लेंगे। यदि हमें यह कहा जाये कि प्रथम 10 प्राकृत संख्याओं का जोड़ निकालना है तो इस समस्या कों हल करने के लिए हमें किसी भी इनपुट वेरियवल की जरूरत

नहीं है।

1.10.3 स्टेप बाई स्टेप हल निकालना (Step by Step Solution)

इनपुट का सही तरीके से पता करने के बाद, अगला कार्य जो भी समस्या है उसका एल्गोरिथम या फ्लोचार्ट बनाना है। जिससे कि उस समस्या से संबंधित हल के सारे स्टेप दर्शाये जायें।

1.10.4 अंकगणितीय एवं तर्क ऑपरेशनों का पता करना(Identification of Arithmetic and logic operations required for solution)

समस्या का सही तरीके से हल निकालने के बाद अगला कार्य यह पता करना है कि किस स्टेप पर अंकगणितीय एवं किस स्टेप पर तर्क ऑपरेशनों का प्रयोग करना है। यह कार्य यदि फ्लोचार्ट बनाया है तो फ्लोचार्ट के बॉक्स के आकार से भी पता चल सकता है।

उदाहरण के लिए कहीं भी यदि निर्णायक बॉक्स (Decision Box) का प्रयोग किया है तो वहां पर तर्क ऑपरेशन लगेगा और यदि प्रोसेसिंग बॉक्स का प्रयोग किया है तो अंकगणितीय ऑपरेशन लगेगा।

1.11 कन्ट्रोल स्ट्रेक्चर (Control Structure)

किसी भी समस्या को सही तरीके से हल करने के लिए चार प्रकार के कन्ट्रोल स्ट्रक्चर होते हैं।

1. अनुक्रम (Sequence)
2. चयन (Selection)
3. लूपिंग (Looping)
4. ब्रॉचिंग (Branching)

1.12 लूपिंग :–

लूपिंग स्ट्रक्चर ज्यादातर जब प्रयोग किया जाता है, जब एक ही कथन एक प्रोग्राम में कई बार प्रयोग आ रहा है। लूपिंग तीन तरह की होती है।

(i) व्हाइल (While)
(ii) डू–व्हाइल (do-While)
(iii) फॉर (for)

1.12.1 व्हाइल लूप :–

व्हाइल लूप किसी समस्या को यूजर द्वारा इच्छित समय तक निष्पादन करने के लिए काम में आती है।

```
Step 1.   n ← 1
Step 2.   while n <10
Step 3.   x ← a * (1+r/100)
Step 4.   write x,n
Step 5.   n ← n+1
Step 6.   end while
```

1.12.2 डू–व्हाइल लूप :–

डू–व्हाइल लूप, व्हाइल लूप की तरह ही काम करती है अंतर केवल इतना ही है कि इसमें शर्त की जाँच बाद में करते हैं जबकि व्हाइल में पहले करते है निम्न उदाहरण मे बोनस के मान की गणना की गई है। जिसमें प्रारम्भिक सैलेरी 2000 अंत सैलेरी 4000 एवं अन्तराल 100 रू. है। बोनस सैलेरी का 20 प्रतिशत होगा।

```
Step 1.   Salary ← 2000
Step 2.   do
          bonus ←.02xSalary
          write bouns
Step 3.   Salary ← Salary +100
Step 4.   while salary <=4000
Step 5.   end
```

1.12.3 फोर लूप :–

फोर लूप साधारणतया जब प्रयोग करते हैं जब निष्पादित करने के पहले ही आपको पता हो कि यह लूप कितनी बार निष्पादित होगी।

1.13 एक अच्छे प्रोग्राम की विशेषताऐं (Characteristics of a good program)

1. रिलायेबल (Reliable)
2. इफिसियन्सी (Efficiency)
3. रीडेबिलीटी (Readability)
4. मैंटेनेबिलिटी (Maintainability)
5. पोर्टेबेलिटी (Portability)

1.13.1 रिलायेबल (Reliable) :- रिलायेबल का मतलब है कि एक प्रोग्राम इस तरह का होना चाहिए, जो किसी भी परिस्थिति में काम करता हो। और यह भी माना जाता है कि प्रोग्राम का परिणाम शुद्ध होगा। जितनी ज्यादा प्रोग्राम की शुद्धता होगी उतनी ज्यादा ही प्रोग्राम रिलायेबल होगा। एक प्रोग्राम जिसका कि परिणाम सही व शुद्ध नहीं हो वह प्रोग्राम रिलायेबल नहीं है।

1.13.2 इफिसियन्सी (Efficiency) :- एक प्रोग्राम को लिखने के कई तरीके हो सकते है। लेकिन इफिसियन्ट प्रोग्राम वह होता है जो कि कम जगह लेता है। तथा जो कि निष्पादन (Executive) में कम समय लगाता है। इस तरह के प्रोग्राम के द्वारा कम्प्यूटर का मूल्य कम हो जाता है, क्योंकि कम्प्यूटर की मेमोरी व समय दोनों ही मँहगे होते है।

1.13.3 रीडेबलिटी (Readability)

प्रोग्राम की सबसे प्रमुख विशेषता रीडेबलिटी होती है। रीडेबलिटी का मतलब है कि एक प्रोग्राम इस तरह का लिखा होना चाहिए कि कोई भी व्यक्ति प्रोग्राम को आसानी से समझ सके।

1.13.4 मैंटेनेबिलीटी (Maintainability)

साधारणतया प्रोग्राम समय और शर्तों के अनुसार परिवर्तन होता रहता है। अब उसी प्रोग्राम को

नये सिरे से दुबारा लिखना कठिन व ज्यादा समय लगने वाला होता है। अतः इस परिस्थिति से निपटने के लिए हमें प्रोग्राम इस तरह का बनाना चाहिये जिससे कि उस प्रोग्राम में परिवर्तन करना आसान हो।

1.13.5 पोर्टेबिलीटी (Portability)

एक पोर्टेबल प्रोग्राम वह होता है जिसे किसी भी कम्प्यूटर पर रन कर सकते हो।

एक अच्छे प्रोग्राम होने के लिए उपरोक्त सारी विशेषताएं होनी चाहिये।

महत्वपूर्ण बिन्दु

1. किसी प्रोग्राम के हल को अलग अलग स्टेप में दिखाने को एल्गोरिथम कहते है।
2. माड्यूलर प्रस्ताव में बडे प्रोग्राम को छोटे–छोटे प्रोग्रामों में बांट देंगे।
3. रन टाइम त्रुटि, प्रोग्राम को रन करते समय आती है।
4. बग का मतलब है त्रुटि और डी बग का मतलब हैं। त्रुटि को हटाना।
5. एक अच्छे प्रोग्राम की विशेषताऐं होती है।

   (i) रिलायेवल (Reliable) (ii) इफिसियन्सी (Efficiency) (iii) रीडेवलीटी (Readability)
   (iv) मैंटेनेबिलिटी (Mantainability) (v) पोर्टेबिलिटी (Portability)

अभ्यासार्थ प्रश्न

बहुचयनात्मक प्रश्न

1. डिवाइड बाई जीरो (Divide by zero) त्रुटि है।

   (अ) सिन्टेक्स त्रुटि (Syntax error) (ब) लॉजीकल त्रुटि (Logical error)

   (स) रन टाइम त्रुटि (Run Time error) (द) उपरोक्त में से कोई नहीं

2. सिन्टेक्स त्रुटि (Syntax error) का पता किया जाता है।

   (अ) रन करते समय (ब) कम्पाइल करते समय

   (स) प्रोग्राम बनाते समय (द) कभी भी नहीं

लघुत्तरात्मक प्रश्न

1. माड्यूल क्या होता है ?
2. पोर्टेबिलिटी से आप क्या समझते हैं ?
3. बग क्या होता है ?
4. एक अच्छे प्रोग्राम की विशेषताएँ बताइये ?

निबन्धात्मक प्रश्न

1 किन्हीं भी तीन संख्याओं का औसत निकालने का एल्गोरिथम बनाइये ?
2. ऐरर कितने प्रकार की होती है ?

उत्तरमाला

1. (स) 2 (ब),