کمپیوٹیشنل تھنکنگ

7

Computational Thinking

سوال 1: کمپیوٹیشنل تھنکنگ کیا ہے، اس کے بنیادی اجزا کون سے ہیں اور یہ مسائل کے حل میں کیسے معاون ثابت ہوتی ہے؟

جواب:

کمپیوٹیشنل تھنکنگ کا تعارف

کمپیوٹیشنل تھنکنگ ایک اہم مہارت ہے جو افراد کو پیچیدہ مسائل کو حل کرنے کے قابل بناتی ہے اور یہ ان طریقوں کا استعمال کرتی ہے جو کمپیوٹر سائنس کے عمل سے مشابہت رکھتے ہیں۔

کمپیوٹیشنل تھنکنگ کے بنیادی اجزا

یہ باب کمپیوٹیشنل تھنکنگ کی تعریف کرنے اور اسے اس کے بنیادی اجزا میں تقسیم کرنے سے شروع ہوتا ہے، جیسے:

(i) تقسیم (Decomposition) (ii) تجزیہ (Analysis)

(iii) پیٹرن کی شناخت (Pattern Recognition) (iv) تجدید (Abstraction)

(v) الگورتھم (Algorithms)

یہ اجزا پیچیدہ مسائل کو آسان بنانے، پیٹرن کی نشاندہی کرنے، غیر ضروری مسائل کو نظر انداز کرنے اور مرحلہ وار طریقہ کار تشکیل دینے کے لیے ضروری ہیں۔

کمپیوٹیشنل تھنکنگ کی عملی اہمیت

ان تصورات کو سمجھنا نہ صرف کمپیوٹر سائنس دانوں کے لیے مفید ہے بلکہ ہر اس شخص کے لیے بھی فائدہ مند ہے جو مختلف شعبوں میں اپنے مسائل کو حل کرنے کی مہارت کو بہتر بنانا چاہتا ہے۔

کمپیوٹیشنل تھنکنگ کے اصول

یہ باب ان اصولوں کا بھی جائزہ لیتا ہے جو رہنمائی فراہم کرتے ہیں، جیسے:

(i) مسئلے کو بہتر طریقے سے سمجھنا (ii) اسے زیادہ منظم اور آسان بنانا

(iii) بہترین حل اور ڈیزائن کا انتخاب کرنا

الگورتھم ڈیزائن کے طریقے

اس باب میں الگورتھم ڈیزائن کرنے کے مختلف طریقے متعارف کرائے گئے ہیں، بشمول:

(i) فلوچارٹ (Flowchart) (ii) سوڈوکوڈ (Pseudocode)

یہ باب ان دونوں طریقوں کے درمیان فرق کی وضاحت بھی کرتا ہے۔

الگورتھم کی عملی مشق

یہ باب LARP (Logical Approach for Resolution of Problems) جیسی سرگرمیوں کے ذریعے الگورتھم ڈیزائن

اور تشخیص کی مہارتوں پر مشق کرنے کی اہمیت پر زور دیتا ہے۔

غلطی کی شناخت اور ڈیبگنگ

آخر میں، باب غلطی کی شناخت اور ڈیبگنگ کے ضروری پہلوؤں کا احاطہ کرتا ہے۔ اس میں الگورتھم کے نفاذ کے دوران پیش آنے والی عام غلطیوں کو پہچاننے اور انھیں ٹھیک کرنے کے طریقے فراہم کیے گئے ہیں۔

سوال 2: کمپیوٹیشنل تھنکنگ کیا ہے اور یہ کن شعبوں میں استعمال کی جاسکتی ہے؟

کیا آپ جانتے ہیں؟
کمپیوٹیشنل تھنکنگ صرف کمپیوٹر سائنس تک محدود نہیں ہے۔ یہ روزمرہ کے مسائل کو حل کرنے میں استعمال ہوتی ہے، جیسے ایک سفر کی منصوبہ بندی کرنا یا کاموں کو ترتیب دینا۔

جواب:

کمپیوٹیشنل تھنکنگ کی تعریف

کمپیوٹیشنل تھنکنگ (CT) کسی مسئلے کو حل کرنے کا ایک ایسا عمل ہے جس میں مہارتوں اور تکنیکوں کا ایک مجموعہ شامل ہوتا ہے، تاکہ پیچیدہ مسائل کو اس انداز میں حل کیا جاسکے جس پر کمپیوٹر بھی عمل درآمد کر سکے۔

کمپیوٹیشنل تھنکنگ کی وسعت

یہ طریقہ کار نہ صرف کمپیوٹر سائنس میں بلکہ دیگر مختلف شعبوں جیسے بیالوجی، ریاضی اور روزمرہ زندگی میں بھی استعمال کیا جاسکتا ہے، جہاں مسائل کو منطقی اور منظم طریقے سے حل کرنے کی ضرورت پیش آتی ہے۔

سوال 3: تقسیم یا تجزیہ (Decomposition) کیا ہے اور پرندوں کے گھر کی تعمیر میں اس کا اطلاق کیسے کیا جاسکتا ہے؟

جواب:

تقسیم یا تجزیہ (Decomposition)

تقسیم ایک پیچیدہ مسئلے کو چھوٹے، زیادہ منظم حصوں میں تقسیم کرنے کا عمل ہے۔ مثال کے طور پر، سکول کی تقریب کی منصوبہ بندی کرتے وقت، کام کو مختلف حصوں میں تقسیم کیا جاسکتا ہے، جیسے تاریخ کا انتخاب، مقام کی بکنگ، دعوت نامے بھیجنا اور سرگرمیوں کا اہتمام کرنا۔

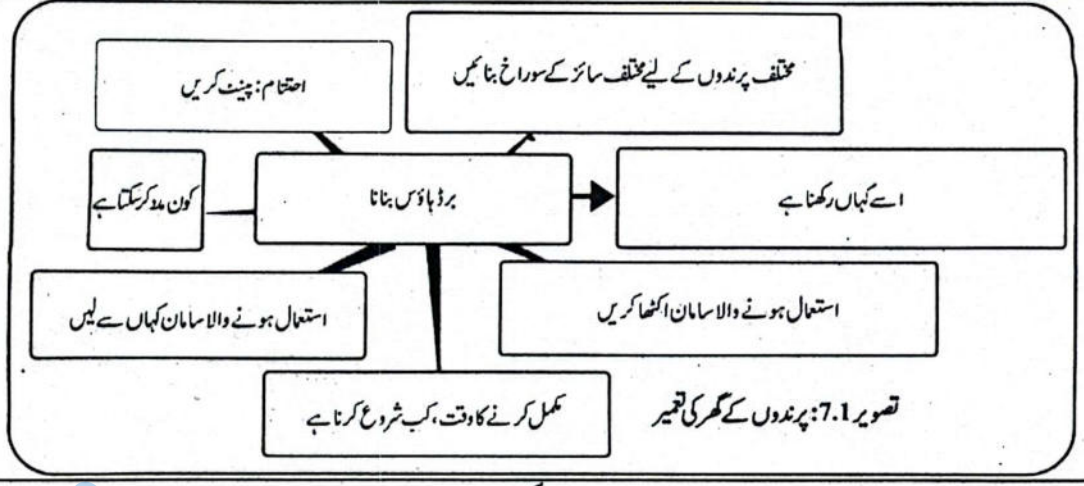
کمپیوٹیشنل تھنکنگ میں تقسیم کی اہمیت

کمپیوٹیشنل تھنکنگ میں تقسیم ایک اہم قدم ہے۔ اس میں پیچیدہ مسائل کو چھوٹے اور قابل انتظام حصوں میں تقسیم کیا جاتا ہے تاکہ ان کا حل آسان بنایا جاسکے۔ مثال کے طور پر، اگر ہمیں پرندوں کے گھر کی تعمیر کا کام کرنا ہو تو ابتدا میں یہ مشکل لگ سکتا ہے، لیکن اگر ہم اسے چھوٹے مراحل میں تقسیم کریں تو ایک وقت میں ایک مرحلہ مکمل کیا جاسکتا ہے۔

پرندوں کے گھر کی تعمیر کے مراحل

تصویر 1.7 میں پرندوں کے گھر کی تعمیر کے لیے تقسیم کے عمل کو ظاہر کیا گیا ہے۔

1- برڈ ہاؤس ڈیزائن کریں: سائز، شکل اور ڈیزائن پر فیصلہ کریں، ایک منصوبہ بنائیں اور تمام ضروری پیمائش جمع کریں۔
2- مواد جمع کریں: لکڑی، کیل، پینٹ، ہتھوڑے اور آری جیسے اوزار سمیت تمام ضروری مواد کی فہرست بنائیں۔
3- لکڑی کو کاٹیں: ڈیزائن کے مطابق لکڑی کی پیمائش کریں اور مطلوبہ حصوں میں کاٹیں۔
4- حصوں کو جوڑنا: پرندوں کے گھر کی ساخت بنانے کے لیے لکڑی کے حصوں کو جوڑنے کے منصوبے پر عمل کریں۔
5- رنگ کریں اور سجائیں: پرندوں کے گھر کو رنگ کریں اور پرندوں کے لیے پرکشش بنانے کے لیے آرائشی اشیا لگائیں۔
6- برڈ ہاؤس نصب کریں: ایک مناسب جگہ تلاش کریں اور محفوظ طریقے سے برڈ ہاؤس نصب کریں جہاں پرندے آسانی سے اس تک پہنچ سکیں۔



تصویر 7.1: پرندوں کے گھر کی تعمیر

سرگرمی
کسی کام کو ڈی کمپوز کریں۔ ایک پیچیدہ کام کے بارے میں سوچیں جو آپ باقاعدگی سے کرتے ہیں، جیسے سکول کی تقریب کا اہتمام کرنا یا کھانا پکانا۔ اسے چھوٹے، قابل انتظام حصوں میں تقسیم کریں۔ ہر مرحلے کو لکھیں اور اپنے ہم جماعتوں کے ساتھ تبادلۂ خیال کریں کہ کس طرح تقسیم کام کو سنبھالنے میں آسان بناتی ہے۔

سوال 4: پیٹرن کی شناخت کمپیوٹیشنل تھنکنگ میں کیوں اہم ہے اور مربعوں کے رقبے میں پیٹرن کیسے دیکھا جاسکتا ہے؟

جواب: پیٹرن کی شناخت (Pattern Recognition)

پیٹرن کی شناخت میں مسائل کے درمیان اور ان کے اندر مماثلت یا نمونے تلاش کرنا شامل ہے۔ مثال کے طور پر، اگر آپ محسوس کرتے ہیں کہ آپ ہمیشہ سوموار کو اپنا ہوم ورک بھول جاتے ہیں، تو یہ ایک پیٹرن ہے اور آپ اتوار کے لیے ایک یاد دہانی ترتیب دے کر اس مسئلے کا حل نکال سکتے ہیں۔

کمپیوٹیشنل تھنکنگ میں پیٹرن کی شناخت کی اہمیت

پیٹرن کی شناخت کمپیوٹیشنل تھنکنگ کا ایک اہم جزو ہے۔ اس میں کسی بھی ڈیٹا یا مسئلے کے اندر ترتیب، باقاعدگی یا تکرار کو سمجھنا شامل ہوتا ہے، جو مسائل کے مؤثر حل میں مدد فراہم کرتا ہے۔

مربعوں کے رقبے میں پیٹرن کی شناخت

آیئے مربعوں کے رقبے میں پیٹرن تلاش کرنے کی ایک مثال دیکھتے ہیں:

مربع کی سائیڈ لمبائی اور رقبہ

تصویر 7.2 میں مربعوں کی سائیڈ کی لمبائی 1 سے 7 تک دکھائی گئی ہے، جبکہ نچلی قطار میں ان کے رقبے دیے گئے ہیں۔ یہاں ایک واضح پیٹرن دیکھا جاسکتا ہے کہ رقبہ میں اضافہ کیسے ہوتا ہے۔

(i) سائیڈ لمبائی 1: رقبہ $1 = 1^2$

(ii) سائیڈ لمبائی 2: رقبہ $(1 + 3)$ $2^2 = 2$

(iii) سائیڈ لمبائی 3: رقبہ $(1 + 3 + 5)$ $3^2 = 9$

(iv) سائیڈ لمبائی 4: رقبہ $(1 + 3 + 5 + 7) = 4^2 = 16$

(v) سائیڈ لمبائی 5: رقبہ $(1 + 3 + 5 + 7 + 9) = 5^2 = 25$

(vi) سائیڈ لمبائی 6: رقبہ $(1 + 3 + 5 + 7 + 9 + 11) = 6^2 = 36$

(vii) سائیڈ لمبائی 7: رقبہ $(1 + 3 + 5 + 7 + 9 + 11 + 13) = 7^2 = 49$

مربع کے رقبے کا حساب

ہم دیکھ سکتے ہیں کہ ہر مربع کے رقبے کا حساب لگاتار طاق نمبروں کو جمع کر کے بھی لگایا جا سکتا ہے۔ مثال کے طور پر، سائیڈ لمبائی 3 کے مربع کا رقبہ حاصل کرنے کے لیے پہلے تین طاق نمبروں کو جمع کیا جاتا ہے:

$$1 + 3 + 5 = 9$$

Visual/Numerical Pattern

Goes up by 1

	+1	+1	+1	+1	+1	+1	
Side	1	2	3	4	5	6	7
Area	1	4	9	16	25	36	47
	+3	+5	+7	+9	+11	+13	

تصویر 7.2: 1 سے 7 تک اطراف والے مربعوں کے رقبہ کا پیٹرن

کیا آپ جانتے ہیں؟
1 سے 10 تک سائیڈ لمبائی کے ساتھ ایک ٹیبل بنائیں۔ لگاتار طاق نمبروں کو جمع کرنے کے پیٹرن کا استعمال کرتے ہوئے مربعوں کے رقبوں کا حساب لگائیں۔ سائیڈ کی لمبائی کو ترتیب دے کر اپنے نتائج کی تصدیق کریں اور دیکھیں کہ آیا پیٹرن برقرار ہے یا نہیں۔

سوال 5: تجدید (Abstraction) کیا ہے اور یہ مسئلہ حل کرنے میں کس طرح مدد دیتی ہے؟

جواب: تجدید (Abstraction)

تجدید مسئلہ حل کرنے میں، خاص طور پر کمپیوٹر سائنس میں، ایک بنیادی تصور ہے۔ اس میں پیچیدہ مسائل کو چھوٹے، زیادہ منظم حصوں میں تقسیم کر کے آسان بنایا جاتا ہے اور غیر ضروری حصوں کو نظر انداز کرتے ہوئے صرف ضروری تفصیلات پر توجہ مرکوز کی جاتی ہے۔ یہ طریقہ مسائل کو زیادہ مؤثر طریقے سے سمجھنے، ڈیزائن کرنے اور حل کرنے میں مدد فراہم کرتا ہے۔

تجدید کی تعریف

تجدید ایک ایسا عمل ہے جس میں پیچیدہ حقیقت کو چھپا کر صرف ضروری عناصر کو نمایاں کیا جاتا ہے۔ اس سے ہمیں تفصیلات میں الجھے بغیر مسائل کا ایک اعلیٰ سطح کا جائزہ لینے اور پیچیدگی کو کم کرنے میں مدد ملتی ہے۔

تجدید کی عملی مثال

چائے کا کپ بنانے کے عمل کو دیکھتے ہیں:

1- پانی ابالیں۔

2- چائے کی پتی یا چائے کا بیگ شامل کریں۔

3- چند منٹ کے لیے دم دینے دیں۔

4- ایک کپ میں ڈالیں اور اگر چاہیں تو دودھ اور چینی شامل کریں۔

دلچسپ معلومات:
یہ پیچیدہ مسائل کو حل کرتے وقت انھیں چھوٹے حصوں میں توڑنے کی کوشش کریں اور اہم مراحل پر توجہ مرکوز کریں۔ یہ آپ کو مسئلے کو بہتر طور پر سمجھنے میں مدد دے گا اور حل تلاش کرنا زیادہ آسان بنائے گا۔ اب straction کا استعمال کرتے ہوئے، ہم پیچیدہ مسائل کا سامنا کر سکتے ہیں۔

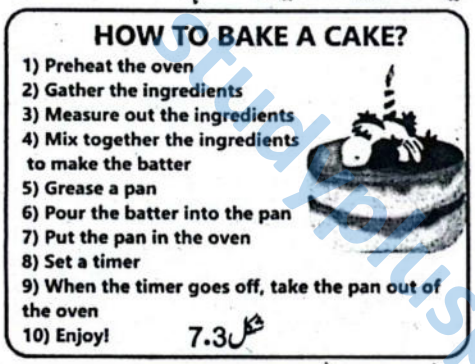
تجرید کا بنیادی اصول

یہ مثال دکھاتی ہے کہ ہم غیر ضروری تفصیلات، جیسے چائے کے برتن کی قسم، پانی کی مقدار، یا دیگر جزوی معلومات کو نظر انداز کر کے صرف بنیادی مراحل پر توجہ دیتے ہیں۔ یہی تجرید کا بنیادی اصول ہے۔

سوال 6: الگورتھم کیا ہوتا ہے، کیک بنانے اور درخت لگانے کے عمل کو ایک الگورتھم کی صورت میں کیسے بیان کیا جا سکتا ہے؟

جواب: الگورتھمز (Algorithms)

الگورتھم کسی مسئلے کو حل کرنے یا کسی کام کو مکمل کرنے کے لیے مرحلہ وار ہدایات کا ایک مجموعہ ہوتا ہے، بالکل ایک کیک پکانے کی ترکیب کی طرح۔ یہ ایک منظم اور واضح ترتیب میں دی گئی ہدایات پر مشتمل ہوتا ہے، جس پر عمل کر کے کسی خاص مقصد کو حاصل کیا جا سکتا ہے۔ الگورتھم بنیادی طور پر ایک ترکیب یا ہدایات کا ایک سیٹ ہوتا ہے جو کسی کام کو انجام دینے کے طریقہ کار کو بیان کرتا ہے۔



شکل 7.3

الگورتھم کی عملی مثالیں

مثال 1: کیک بنانا

تصویر میں ہم کیک بنانے کی ترکیب دیکھ سکتے ہیں۔ ترکیب میں اجزا کی ایک فہرست فراہم کی جاتی ہے اور انھیں ملانے اور کیک پکانے کے لیے مرحلہ وار ہدایات دی جاتی ہیں۔ یہ الگورتھم کی ایک مثال ہے کیونکہ یہ کیک بنانے کے مقصد کو حاصل کرنے کے لیے اقدامات کی ایک واضح ترتیب فراہم کرتا ہے۔

مثال 2: درخت لگانے کا الگورتھم

یہاں درخت لگانے کے لیے ایک آسان الگورتھم دیا گیا ہے، جو ایک مفید اور بامعنی سرگرمی ہو سکتی ہے:

1- اپنے باغ میں ایک مناسب جگہ کا انتخاب کریں۔
2- ایک گڑھا کھودیں جو درخت کی جڑ سے دوگنا چوڑا ہو۔
3- درخت کو گڑھے میں رکھیں اور یقینی بنائیں کہ درخت سیدھا ہے۔
4- گڑھے کو مٹی سے بھریں اور ہوا کو ہٹانے کے لیے اسے آہستہ سے دبائیں۔
5- درخت کو دل کھول کر پانی دیں تا کہ وہ مٹی میں جڑ پکڑ سکے۔
6- نمی کو برقرار رکھنے کے لیے درخت کی جڑوں کے اردگرد نامیاتی کھاد ڈالیں۔
7- درخت کو باقاعدگی سے پانی دیں جب تک کہ وہ بڑھنا شروع نہ ہو جائے۔

واضح ہدایات

یہ الگورتھم درخت لگانے کے بارے میں واضح ہدایات فراہم کرتا ہے، جس سے کسی کے لیے بھی اس پر عمل کرنا آسان ہو جاتا ہے۔

سرگرمی

آئیں ایک الگورتھم بنائیں!

کسی ایسے کام کے بارے میں سوچیں جو آپ ہر روز کرتے ہیں، جیسے دانتوں کو برش کرنا یا سکول بیگ کو پیک تیار کرنا۔ آپ جن اقدامات پر عمل کرتے ہیں انھیں ایک ایک کر کے لکھیں۔ اپنی کلاس کے ساتھ اپنے الگورتھم کا اشتراک کریں اور دیکھیں کہ آیا آپ کے دوست اس پر عمل کر سکتے ہیں!

کیا آپ جانتے ہیں؟
الگورتھم صرف کمپیوٹرز میں ہی استعمال نہیں ہوتے ہیں؟ وہ ہر جگہ موجود ہیں۔ جب آپ اپنے دوست کے گھر جانے کے لیے ہدایات پر عمل کرتے ہیں یا قواعد کے ساتھ بورڈ گیم کھیلتے ہیں تو آپ الگورتھم استعمال کر رہے ہوتے ہیں۔ الگورتھم ہمیں منطقی طریقے سے مسائل کو حل کرنے میں مدد کرتے ہیں۔

سرگرمی
جماعت نہم 9 کے امتحان کے لیے بورڈ آف انٹرمیڈیٹ اینڈ سیکنڈری ایجوکیشن (BISE) میں درخواست دینے کے لیے ایک الگورتھم کا خاکہ تیار کریں۔

سوال 7: طلبہ کس طرح ایک عام کام کے لیے الگورتھم تیار کر کے اس کی مؤثریت کو جانچ سکتے ہیں؟

جواب:

الگورتھم چیلنج (Algorithm Challenge)

الگورتھمز عملی زندگی میں مختلف کاموں کو منظم اور آسان بنانے میں مدد دیتے ہیں۔ اس چیلنج میں، طلبہ کو ایک عام کام کے لیے الگورتھم بنانے کا تجربہ دیا جاتا ہے تاکہ وہ مرحلہ وار ہدایات کی اہمیت کو سمجھ سکیں۔

گروپ میں کام کرنا

طلبہ دو دو کے گروپوں میں کام کریں اور کسی عام کام، جیسے سینڈوچ بنانا یا سکول کے لیے تیار ہونا، کے لیے الگورتھم تیار کریں۔ ہر مرحلے کو واضح طور پر لکھیں تاکہ کوئی اور شخص بھی انھی ہدایات پر عمل کر کے کام مکمل کر سکے۔

الگورتھم کا تبادلہ اور جانچ

اپنا تیار کردہ الگورتھم کسی دوسرے گروپ کے ساتھ تبادلہ کریں۔ دوسرے گروپ کے الگورتھم پر بالکل اسی طرح عمل کریں جیسے لکھا گیا ہے اور دیکھیں کہ آیا آپ کام صحیح طریقے سے مکمل کر سکتے ہیں۔ اس سے معلوم ہوگا کہ الگورتھم واضح اور مؤثر ہے یا اس میں بہتری کی ضرورت ہے۔

سوال 8: مسئلے کی تفہیم کمپیوٹیشنل تھنکنگ میں کیوں اہم ہے اور اس کی وضاحت کے لیے سکول کی ویب سائٹ بنانے کی مثال کیسے مددگار ثابت ہو سکتی ہے؟

جواب:

کمپیوٹیشنل تھنکنگ کے اصول

کمپیوٹیشنل تھنکنگ میں کئی اہم اصول شامل ہیں جو منظم انداز میں مسئلہ حل کرنے کے عمل کی رہنمائی کرتے ہیں۔

مسئلے کی تفہیم

کسی مسئلے کو سمجھنے میں بنیادی مسئلے کی شناخت، ضروریات کی وضاحت اور مقاصد کا تعین شامل ہوتا ہے۔ یہ مسئلہ حل کرنے کا پہلا اور سب سے اہم مرحلہ ہے، خاص طور پر کمپیوٹیشنل تھنکنگ میں۔ اس میں حل تلاش کرنے کی کوشش سے پہلے اس کے بنیادی اجزا اور ضروریات کی شناخت کے لیے مسئلے کا مکمل تجزیہ کرنا شامل ہے۔

البرٹ آئن سٹائن کا قول:

''اگر میرے پاس کسی مسئلے کو حل کرنے کے لیے ایک گھنٹا ہوتا تو میں 55 منٹ اس مسئلے کے بارے میں سوچنے میں گزارتا اور 5 منٹ اس کے حل کے بارے میں سوچنے میں صرف کرتا۔''

مسئلے کی تفہیم کی اہمیت

1- وضاحت اور توجہ:

مسئلے کو مکمل طور پر سمجھنے سے آپ کو معلوم ہو جاتا ہے کہ کیا حل کرنے کی ضرورت ہے۔ اس سے آپ غیر ضروری تفصیلات میں الجھنے کی بجائے اصل مسئلہ پر توجہ مرکوز کر سکتے ہیں۔

2- اہداف کا تعین:

مسئلے کی مناسب سمجھ سے آپ واضح اور قابل حصول اہداف طے کر سکتے ہیں۔ آپ اس بات کا تعین کر سکتے ہیں کہ حتمی نتیجہ کیسا ہونا چاہیے اور اس نتیجے تک پہنچنے کے لیے کون سے مخصوص اقدامات ضروری ہیں۔

3- مؤثر حل:

جب آپ مسئلے کو اچھی طرح سمجھتے ہیں تو آپ زیادہ مؤثر حل بنا سکتے ہیں۔ آپ مسئلے کو حل کرنے میں وقت اور وسائل کی بچت کرنے کے لیے بہترین طریقوں اور آلات کا انتخاب کر سکتے ہیں۔

4- غلطیوں سے بچنا:

مسئلے کو اچھی طرح سمجھ کر آپ عام نقصانات اور غلطیوں سے بچ سکتے ہیں۔ مسئلے کو غلط سمجھنا غلط نتائج اور محنت کے ضائع ہونے کا باعث بن سکتا ہے۔

مثال: سکول کی ویب سائٹ بنانا

فرض کریں کہ آپ کو اپنے سکول کے لیے ایک ویب سائٹ بنانے کے لیے کہا جاتا ہے۔ کوڈنگ میں جانے سے پہلے، آپ کو مسئلہ سمجھنے کی ضرورت ہے:

1- ضروریات کی شناخت:

ویب سائٹ میں کن خصوصیات کی ضرورت ہے؟ مثال کے طور پر، خبروں، واقعات، کلاس شیڈول اور رابطے کی معلومات کے لیے صفحات۔

2- صارف کی ضروریات:

ویب سائٹ کا استعمال کون کرے گا؟ طالب علم، اساتذہ یا والدین؟ اپنے سامعین کو سمجھنے سے ایک صارف دوست انٹرفیس ڈیزائن کرنے میں مدد ملے گی۔

3- تکنیکی رکاوٹیں:

کون سے وسائل اور آلات دستیاب ہیں؟ کیا آپ کو ویب سرور اور ضروری سافٹ ویئر تک رسائی حاصل ہے؟

ان تمام پہلوؤں کو سمجھ کر، آپ ایک ویب سائٹ کی منصوبہ بندی اور تعمیر کر سکتے ہیں جو آپ کے سکول کی ضروریات کو مؤثر طریقے سے پورا کرے۔

دلچسپ معلومات:
کسی مسئلے کو حل کرنا شروع کرنے سے پہلے ہمیشہ اسے اچھی طرح سے سمجھنے کے لیے وقت نکالیں۔ سوالات پوچھیں، معلومات جمع کریں اور کسی بھی شک کی وضاحت کریں۔ یہ بنیادی قدم بہتر اور زیادہ مؤثر حل کا باعث بنے گا۔

سوال 9: مسئلے کو آسان بنانے اور بہترین حل کے انتخاب میں کون سے اقدامات شامل ہوتے ہیں؟

جواب:

مسئلے کو آسان بنانا

کسی مسئلے کو آسان بنانے کا مطلب ہے کہ اسے چھوٹے، زیادہ قابلِ انتظام ذیلی مسائل میں تقسیم کیا جائے۔ اس طریقے سے بڑے اور پیچیدہ مسائل کو مرحلہ وار حل کرنا ممکن ہوتا ہے۔

مثال: ویب سائٹ ڈیزائن کرنے کے عمل کو چھوٹے کاموں میں تقسیم کیا جا سکتا ہے:

(i) لے آؤٹ ڈیزائن کرنا۔ ویب سائٹ کے بنیادی ڈھانچے اور ترتیب کو طے کرنا۔

(ii) مواد بنانا۔ ویب سائٹ کے صفحات کے لیے متن، تصاویر اور دیگر ضروری معلومات تیار کرنا۔

(iii) کوڈنگ۔ ویب سائٹ کے مختلف عناصر کو عملی شکل دینا اور فنکشنل بنانا۔

حل کا انتخاب اور ڈیزائن

بہترین حل کا انتخاب کرنے میں مختلف طریقوں کا جائزہ لینا اور سب سے زیادہ مؤثر طریقے کا انتخاب کرنا شامل ہے۔ یہ یقینی بناتا ہے کہ مسئلے کا حل نہ صرف درست ہو بلکہ وسائل اور وقت کے لحاظ سے بھی موزوں ہو۔

حل کے ڈیزائن کا عمل:

(i) مختلف حلوں کا جائزہ لینا۔ دستیاب طریقوں کا موازنہ کر کے سب سے بہتر آپشن کا انتخاب کرنا۔

(ii) منصوبہ بندی کرنا۔ حل کو عملی جامہ پہنانے کے لیے ایک تفصیلی حکمت عملی تیار کرنا۔

(iii) الگورتھم بنانا۔ منتخب کردہ حل کے مطابق قدم بہ قدم ہدایات ترتیب دینا تاکہ مسئلے کا حل مؤثر اور منظم ہو۔

سوال 10: الگورتھم ڈیزائن میں فلوچارٹ کیا ہوتا ہے، اس کی کیا اہمیت ہے اور اس میں استعمال ہونے والی مختلف علامتوں کی وضاحت کریں؟

جواب:

الگورتھم ڈیزائن کرنے کے طریقے

الگورتھم ڈیزائن کے مختلف طریقے کمپیوٹیشنل مسائل سے مؤثر طریقے سے نمٹنے کے لیے مختلف ٹولز اور تکنیک فراہم کرتے ہیں۔ ہر طریقہ کار کی اپنی طاقت اور کمزوریاں ہوتی ہیں، جو اسے مخصوص مسائل کے لیے موزوں بناتی ہیں۔ مختلف طریقوں کو سمجھنے سے بہترین حل کا انتخاب کرنے میں مدد ملتی ہے۔ آیئے دو اہم طریقوں پر بات کرتے ہیں۔

فلوچارٹ

فلوچارٹ کسی عمل یا نظام کے مراحل کی بصری نمائندگی فراہم کرتے ہیں، جو تیروں سے منسلک مختلف علامتوں کا استعمال کرتے ہوئے بنائے جاتے ہیں۔ یہ وسیع پیمانے پر کمپیوٹر سائنس، انجینئرنگ اور کاروبار جیسے شعبوں میں استعمال ہوتے ہیں تاکہ مختلف عمل کو واضح اور مؤثر انداز میں بیان کیا جا سکے۔

فلوچارٹ کی اہمیت

(i) **وضاحت:** فلوچارٹ کسی بھی عمل کو واضح اور مختصر انداز میں پیش کرتے ہیں، جس سے اسے ایک نظر میں سمجھنا آسان ہو جاتا ہے۔

(ii) **ابلاغ:** یہ پیچیدہ عمل کو وسیع سامعین تک پہنچانے کے لیے بہترین ذریعہ ہیں تاکہ سب کو ایک جیسی سمجھ حاصل ہو۔

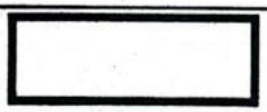
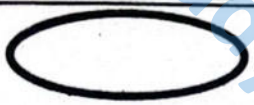
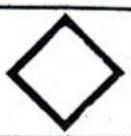
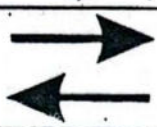
(iii) **مسئلہ حل کرنا:** فلوچارٹ کسی عمل میں رکاوٹوں اور غیر مؤثر حصوں کی نشاندہی کرنے میں مدد کرتے ہیں، جس سے مسئلہ حل کرنے اور بہتری

لانے میں آسانی ہوتی ہے۔

(iv) **دستاویزات:** فلوچارٹ نظام اور عمل کے لیے ایک ضروری دستاویز کے طور پر کام کرتے ہیں، جو تربیت اور حوالہ کے مقاصد کے لیے مفید ہوتے ہیں۔

فلوچارٹ کی علامتیں

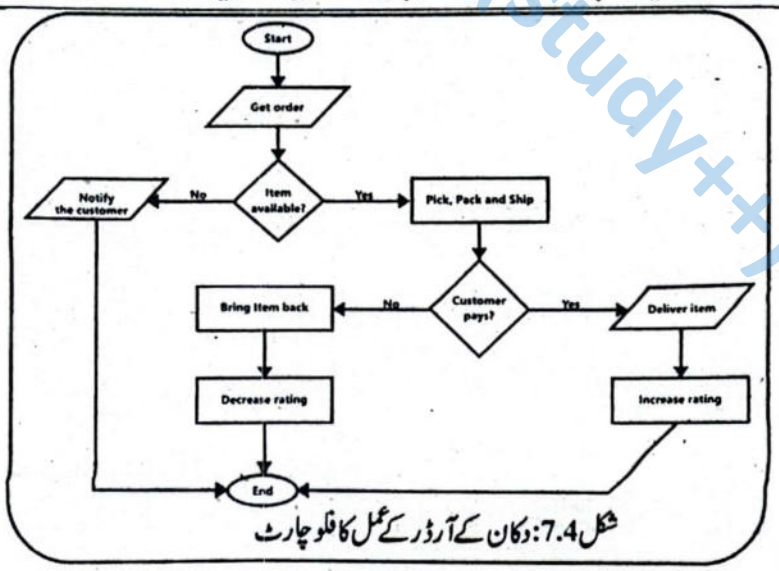
فلوچارٹ میں مختلف علامتیں استعمال کی جاتی ہیں جو کسی عمل یا نظام کے مراحل اور بہاؤ کی وضاحت کے لیے کام آتی ہیں۔

بیان	نام	علامت
کسی عمل کے آغاز یا اختتام کی نمائندگی کرتا ہے۔ اکثر لیبل لگایا جاتا ہے۔ "Start" یا "End"	بیضوی (ٹرمینل)	
ایک عمل، کام یا آپریشن کی نمائندگی کرتا ہے جسے انجام دینے کی ضرورت ہے۔	مستطیل (عمل)	
ڈیٹا ان پٹ یا آؤٹ پٹ کی نمائندگی کرتا ہے (مثال کے طور پر صارف متوازی گرام سے ان پٹ حاصل کرنا یا سکرین پر آؤٹ پٹ دکھانا۔)	متوازی گرام (Input / Output)	
یہ عمل میں ایک فیصلہ کی جگہ کی نمائندگی کرتا ہے جہاں بہاؤ ہاں/ نہیں سوال، یا صحیح/غلط کی بنیاد پر مختلف ہو سکتا ہے۔	ڈائمنڈ (فیصلہ سازی)	
فلو چارٹ کے اندر بہاؤ کی سمت دکھاتا ہے اور اقدامات کی ترتیب کی نشان دہی کرنے کے لیے علامتوں کو جوڑتا ہے۔	تیر (فلولائن)	

ٹیبل 7.1: فلوچارٹ علامتیں

کیا آپ جانتے ہیں؟

فلوچارٹ کو کمپیوٹر سائنس دانوں جیسے کہ جان وان نیومین اور ہرمن گولڈ سٹائن نے کمپیوٹنگ کے ابتدائی دنوں میں مقبول بنایا۔



شکل 7.4: دکان کے آرڈر کے عمل کا فلوچارٹ

سرگرمی
ایک روزمرہ کی سرگرمی کے لیے ایک فلوچارٹ بنائیں، جیسے سکول کے لیے تیار ہونا۔ جیسے اس میں فیصلہ کن نکات شامل کریں۔ موسم کی بنیاد پر کیا پہننا ہے۔

سوال 11: دکان کے آرڈر پروسیسنگ فلوچارٹ میں گاہک کی درجہ بندی کو شامل کرنے سے کیا بہتری حاصل ہوتی ہے؟

جواب:

مثال: آپ کے گھر کے قریب ایک دکان

فرض کریں کہ ایک دکان موبائل فون پیغامات کے ذریعے آرڈر لیتی ہے۔ شکل 7.4 میں دیے گئے فلوچارٹ میں آرڈر پروسیسنگ کے مراحل کا خاکہ پیش کیا گیا ہے۔ اس عمل میں:

ان پٹ: آرڈر وصول کرنا

آؤٹ پٹ: آئٹم کی ڈیلیوری یا اطلاع دینا اگر آئٹم دستیاب نہ ہو

فیصلہ سازی کے مراحل

آرڈر پروسیسنگ میں درج ذیل فیصلے کیے جاتے ہیں:

آئٹم کی دستیابی:

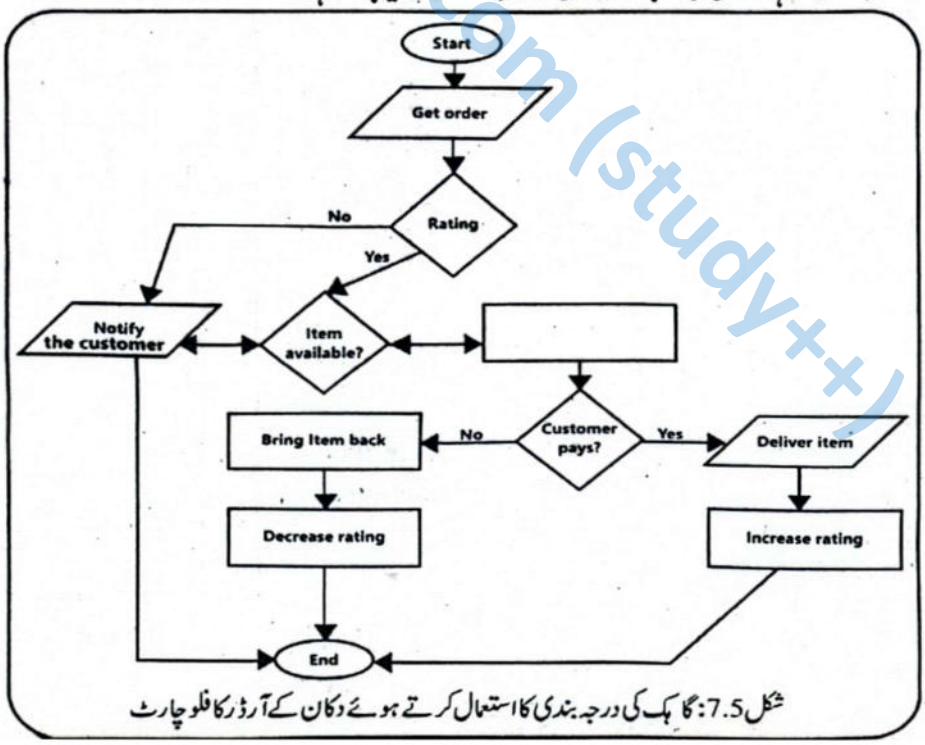
اگر آئٹم دستیاب نہیں ہے، تو دکان گاہک کو مطلع کر دیتی ہے۔

اگر آئٹم دستیاب ہے، تو دکان اسے اٹھاتی، پیک کرتی اور بھیجتی ہے۔

گاہک کی ادائیگی:

اگر گاہک آئٹم قبول نہیں کرتا یا ادائیگی نہیں کرتا، تو آئٹم دکان پر واپس کر دیا جاتا ہے اور گاہک کی درجہ بندی 1 نمبر کم ہو جاتی ہے۔

اگر گاہک ادائیگی کر دیتا ہے، تو اس کی درجہ بندی میں 1 نمبر کا اضافہ کر دیا جاتا ہے۔



شکل 7.5: گاہک کی درجہ بندی کا استعمال کرتے ہوئے دکان کے آرڈر کا فلوچارٹ

گاہک کی درجہ بندی کا استعمال کرتے ہوئے فلوچارٹ میں بہتری

نوٹ کریں کہ اگرچہ گاہک کی درجہ بندی پہلے فلوچارٹ (شکل 7.4) میں شامل ہے، لیکن اس کا کوئی عملی استعمال نہیں کیا گیا تھا۔ اس مسئلے کو حل کرنے کے لیے، فلوچارٹ کو بہتر بنایا گیا ہے (شکل 7.5) تاکہ صرف وہی صارفین آرڈر کر سکیں جن کی درجہ بندی ایک مخصوص حد (مثلاً 3 یا اس سے زیادہ) سے زیادہ ہو۔

یہ بہتری یقینی بناتی ہے کہ صرف مستند اور قابلِ بھروسا صارفین ہی آرڈر دے سکیں، جس سے دکان کے لیے غیر ضروری نقصانات اور غیر سنجیدہ آرڈرز کو کم کیا جا سکتا ہے۔

سرگرمی
شکل 7.5 کو ترمیم کریں تاکہ صارف کی ریٹنگز درست حد 10 سے 5 کے درمیان ہوں بشمول۔ ریٹنگز منفی نہیں ہو سکتی ہیں یا 5 سے زیادہ نہیں ہو سکتی ہیں۔

سوال 12: لاگ ان سسٹم کے فلوچارٹ میں صارف کی تصدیق اور رسائی کے مراحل کیا ہیں؟

جواب:

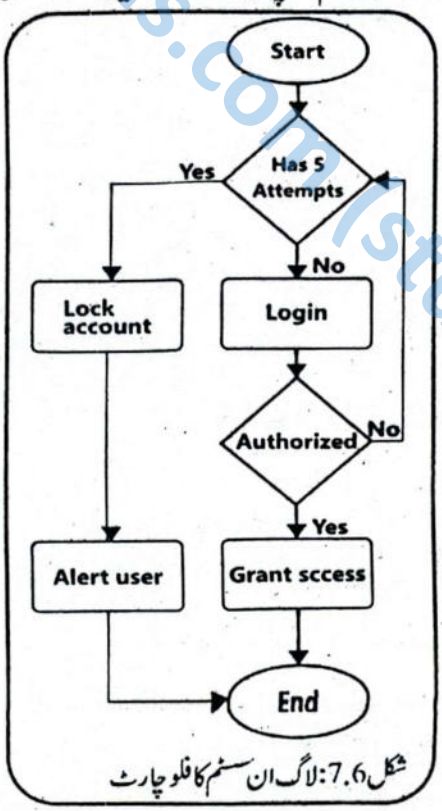
مثال: لاگ ان سسٹم کے لیے فلوچارٹ

یہ فلوچارٹ ایک لاگ ان سسٹم کے عمل کو ظاہر کرتا ہے، جس میں صارف کی اسناد کی تصدیق اور رسائی دینے کے مراحل شامل ہیں۔

لاگ ان کے مراحل

صارف کی معلومات درج کرنا۔ صارف اپنا نام اور پاس ورڈ داخل کرتا ہے۔

سسٹم چیک کرتا ہے کہ آیا داخل کردہ صارف نام اور پاس ورڈ درست ہیں۔۔ اسناد کی تصدیق



شکل 7.6: لاگ ان سسٹم کا فلوچارٹ

رسائی کی اجازت یا انکار

اگر اسناد درست ہیں، تو صارف کو سسٹم تک رسائی دے دی جاتی ہے۔
اگر اسناد غلط ہیں، تو صارف کو دوبارہ کوشش کرنے کی اجازت دی جاتی ہے۔
کوششوں کی حد – صارف زیادہ سے زیادہ پانچ مرتبہ لاگ ان کی کوشش کر سکتا ہے۔ اگر تمام کوششیں ناکام ہو جائیں تو سسٹم مزید کوشش کی اجازت نہیں دیتا اور رسائی بلاک کر دی جاتی ہے۔

بہتری کے پہلو

سیکیورٹی میں اضافہ – غیر مجاز صارفین کو متعدد بار لاگ ان کرنے سے روکنے کے لیے کوششوں کی حد متعین کی گئی ہے۔
بہتر صارف تجربہ – غلط اسناد کی صورت میں صارف کو موقع دیا جاتا ہے کہ وہ درست تفصیلات دوبارہ درج کرے۔

سوال 13: سوڈوکوڈ کیا ہے اور یہ الگورتھم کی منصوبہ بندی میں کیسے مدد دیتا ہے؟

جواب: سوڈوکوڈ (Pseudocode)

سوڈوکوڈ سادہ اور غیر رسمی زبان کا استعمال کرتے ہوئے الگورتھم کو پیش کرنے کا ایک طریقہ ہے، جسے سمجھنا آسان ہوتا ہے۔ یہ پروگرامنگ کی ساخت کو سادہ انگریزی جیسی پڑھنے میں آسان زبان میں بیان کرتا ہے، جس سے الگورتھم کی منصوبہ بندی اور وضاحت کے لیے ایک مؤثر ذریعہ بن جاتا ہے۔

سوڈوکوڈ کیا ہے؟

سوڈوکوڈ اصل میں کوئی ایسا کوڈ نہیں ہوتا جسے کمپیوٹر پر چلایا جا سکے، بلکہ یہ الگورتھم کے مراحل کو اس انداز میں بیان کرتا ہے جسے ہر کوئی آسانی سے سمجھ سکے۔ یہ خاص طور پر پروگرامرز اور طالب علموں کے لیے مفید ہوتا ہے، کیونکہ اس میں کسی مخصوص پروگرامنگ زبان کے قواعد میں الجھے بغیر الگورتھم کی منطق پر توجہ مرکوز کرنے میں مدد ملتی ہے۔

سوال 14: کسی عدد کے جفت یا طاق ہونے کی جانچ کے لیے سوڈوکوڈ کیسے کام کرتا ہے؟

جواب: مثال: عدد کا جفت یا طاق ہونا جانچنے کا طریقہ

کسی عدد کے جفت (Even) یا طاق (Odd) ہونے کا تعین پروگرامنگ اور کمپیوٹر سائنس میں ایک بنیادی کام ہے۔ ایک عدد جفت ہوتا ہے اگر وہ بغیر باقی بچے 2 سے تقسیم ہو جائے، بصورت دیگر وہ طاق ہوتا ہے۔

سوڈوکوڈ برائے جفت یا طاق عدد کی جانچ

ذیل میں دیے گئے سوڈوکوڈ کے ذریعے معلوم کیا جا سکتا ہے کہ کوئی عدد جفت ہے یا طاق:

الگورتھم 1: جفت یا طاق عدد کی جانچ کا سوڈوکوڈ

```
1:    Procedure CheckEvenOdd(number)
2:    Input: number {The number to be checked)
3:    Output: "Even" if number is even, "Odd" if number is odd
4:    Begin
5:        if (number%2 = 0) then
```

```
6:              print "Even"
7:      else
8:              print "Odd"
9:      End if
10:     End
```

وضاحت

(i) پروسیجر کا اعلان:

سوڈو کوڈ CheckEvenOdd نامی پروسیجر سے شروع ہوتا ہے، جو ایک ان پٹ "Number" لیتا ہے۔

(ii) ان پٹ:

طریقہ کار ایک ویری ایبل "Number" کو قبول کرتا ہے، جو چیک کیا جانے والا عدد ہے۔

(iii) آؤٹ پٹ:

اگر عدد جفت (Even) ہے تو پروسیجر "Even" پرنٹ کرے گا۔
اگر عدد طاق (Odd) ہے تو پروسیجر "Odd" پرنٹ کرے گا۔

(iv) آغاز:

Begin کا استعمال پروسیجر کے آغاز کو ظاہر کرتا ہے۔

(v) شرط کی جانچ:

if (Number%2 = 0) چیک کرتا ہے کہ آیا عدد کو 2 سے تقسیم کرنے پر باقی صفر آتا ہے یا نہیں۔ اس مقصد کے لیے موڈیولس آپریٹر (%) استعمال کیا جاتا ہے۔

(vi) Even کیس:

اگر شرط درست ہو، یعنی باقی صفر ہو، تو طریقہ کار "Even" پرنٹ کرتا ہے۔

(vii) Odd کیس:

اگر شرط غلط ہو، یعنی باقی صفر نہ ہو، تو طریقہ کار "Odd" پرنٹ کرتا ہے۔

(viii) اختتام:

End طریقہ کار کے اختتام کو ظاہر کرتا ہے۔

سوال 15: کسی عدد کے مفرد (Prime) ہونے کی جانچ کے لیے استعمال ہونے والے سوڈو کوڈ کے مراحل کون سے ہیں اور یہ کس منطق پر کام کرتا ہے؟

جواب: مثال: کسی عدد کے مفرد (Prime) ہونے کی جانچ

کسی عدد کے مفرد (Prime) ہونے کا تعین نمبر تھیوری اور کمپیوٹر سائنس میں ایک بنیادی کام ہے۔ ایک مفرد عدد وہ قدرتی عدد ہوتا ہے جو 1 سے بڑا ہو اور صرف 1 یا خود پر تقسیم ہو سکتا ہو۔

سوڈوکوڈ برائے مفرد عدد کی جانچ

ذیل میں دیے گئے سوڈوکوڈ کے ذریعے معلوم کیا جا سکتا ہے کہ کوئی عدد مفرد ہے یا نہیں:

الگورتھم 2: مفرد عدد کی جانچ کا سوڈوکوڈ

```
1:    Procedure Is Prime(number)
2:    Input: number {The number to be checked)
3:    Output: True if number is prime, False otherwise
4:    Begin
5:        if(number<=) then
6:            return False
7:        end if
8:        for i fro 2 to sqrt(number) do
9:            if (numer%i = 0) then
10:               return False
11:           end if
12:       end for
13:       return True
14:   End
```

وضاحت

(i) پروسیجر کا اعلان:

سوڈوکوڈ ''isPrime'' نامی پروسیجر سے شروع ہوتا ہے، جو ایک ان پٹ "Number" لیتا ہے۔

(ii) ان پٹ:

طریقہ کار ایک ویری ایبل "Number" کو قبول کرتا ہے، جو چیک کیا جانے والا عدد ہے۔

(iii) آؤٹ پٹ:

اگر عدد مفرد (Prime) ہے تو پروسیجر True واپس کرے گا۔

اگر عدد مفرد نہیں ہے تو پروسیجر False واپس کرے گا۔

(iv) آغاز:

Begin کا استعمال پروسیجر کے آغاز کو ظاہر کرتا ہے۔

(v) ابتدائی چیک:

if (Number < = 1) چیک کرتا ہے کہ آیا عدد 1 سے کم یا برابر ہے۔
اگر یہ شرط درست ہو، تو پروسیجر False واپس کرتا ہے کیونکہ 1 یا اس سے چھوٹے اعداد مفرد نہیں ہوتے۔

(vi) ممکنہ تقسیم کاروں کی تلاش کے لیے لوپ:

for لوپ عدد کے جذر (Square Root) تک چلتا ہے۔
اس کی وجہ یہ ہے کہ کسی عدد کے بڑے عوامل (Factors) اس کے چھوٹے عوامل کے ضرب (Multiplication) سے حاصل کیے جا سکتے ہیں، جنھیں پہلے ہی چیک کیا جا چکا ہوتا ہے۔

(vii) تقسیم کی جانچ:

if(Number % i = 1) چیک کرتا ہے کہ آیا عدد کو i پر پورے طور پر تقسیم کیا جا سکتا ہے۔
اگر یہ شرط True ہو، تو پروسیجر False واپس کرتا ہے کیونکہ اس عدد کا 1 کے علاوہ ایک اور بھی تقسیم کار موجود ہے، یعنی یہ مفرد نہیں ہے۔

(viii) مفرد عدد کی تصدیق:

اگر for لوپ میں کوئی تقسیم کار نہیں پایا جاتا، تو پروسیجر True واپس کرتا ہے، جس سے تصدیق ہوتی ہے کہ عدد مفرد ہے۔

(ix) اختتام:

End طریقہ کار کے اختتام کو ظاہر کرتا ہے۔

سرگرمی
اپنا سوڈو کوڈ بنائیں: طالب علموں کو چھوٹے گروپوں میں تقسیم کریں اور ہر گروپ کو ایک مختلف آسان مسئلہ تفویض کریں، جیسے کسی فہرست میں زیادہ سے زیادہ تعداد تلاش کرنا یا کسی نمبر کی Factorial کا حساب لگانا۔ انھیں اپنے تفویض کردہ مسئلے کے لیے سوڈو کوڈ لکھنے کے لیے کہیں اور پھر اسے کلاس میں پیش کریں۔

کیا آپ جانتے ہیں؟
اصل کوڈ لکھنے سے پہلے سافٹ ویئر کی تیاری میں سوڈو کوڈ کو اکثر استعمال کیا جاتا ہے تا کہ اس بات کو یقینی بنایا جا سکے کہ منطق درست ہے۔ یہ ٹیم کے افراد کے مابین رابطے کو بھی آسان بناتا ہے خاص طور پر جب وہ رابطے مختلف پروگرمنگ لینگویجز استعمال کر رہے ہیں۔

سوال 16: سوڈو کوڈ استعمال کرنے کے کیا فوائد ہیں؟

جواب: سوڈو کوڈ کیوں استعمال کریں؟

سوڈو کوڈ کے استعمال کے متعدد فوائد ہیں، جو الگورتھم کی منصوبہ بندی اور تفہیم کو آسان بناتے ہیں۔

1۔ وضاحت

سوڈو کوڈ کسی مخصوص پروگرامنگ زبان کے ٹیکسٹ کی پیچیدگیوں کے بغیر الگورتھم کی منطق کو واضح طور پر بیان کرنے میں مدد کرتا ہے۔

2۔ منصوبہ بندی

یہ پروگرامرز کو اپنے خیالات کا خاکہ بنانے اور الگورتھم کے مراحل کی بہتر منصوبہ بندی کرنے میں معاون ثابت ہوتا ہے۔

3۔ ابلاغ

سوڈوکوڈ الگورتھم کے مراحل کو ایک عالمگیر انداز میں پیش کرنے کا ذریعہ ہے، جس سے دوسروں کے ساتھ تبادلۂ خیال کرنا اور پیچیدہ تصورات کو سمجھانا آسان ہو جاتا ہے۔

سوال 17: فلوچارٹ اور سوڈوکوڈ میں کیا فرق ہے اور الگورتھم کی وضاحت کے لیے کون سا طریقہ کب زیادہ مؤثر ہوتا ہے؟

جواب:

فلوچارٹ اور سوڈوکوڈ میں فرق

فلوچارٹ اور سوڈوکوڈ دونوں الگورتھم کی وضاحت کے لیے استعمال کیے جاتے ہیں، لیکن یہ مختلف طریقوں سے ایسا کرتے ہیں۔ ان کے درمیان فرق کو سمجھنا اہم ہے تاکہ یہ فیصلہ کیا جا سکے کہ کس موقع پر کون سا طریقہ زیادہ مؤثر ہوگا۔

سوڈوکوڈ

(i) سوڈوکوڈ الگورتھم کے مراحل کی وضاحت کے لیے سادہ زبان اور اسٹرکچرڈ فارمیٹ کا استعمال کرتا ہے۔

(ii) یہ ایک کہانی کی طرح پڑھا جاتا ہے، جہاں ہر مرحلہ ترتیب وار لکھا جاتا ہے۔

(iii) سوڈوکوڈ تفصیلی اور بیانیہ طرز میں الگورتھم کے مراحل کو واضح کرتا ہے۔

(iv) یہ خاص طور پر الگورتھم کی ایسی دستاویز سازی کے لیے مفید ہے، جسے کسی بھی پروگرامنگ زبان میں آسانی سے تبدیل کیا جا سکتا ہے۔

فلوچارٹ

(i) فلوچارٹ الگورتھم کے بہاؤ کو ظاہر کرنے کے لیے گرافیکل علامتوں اور تیروں (arrows) کا استعمال کرتے ہیں۔

(ii) یہ ایک فلم دیکھنے کی طرح ہے، جہاں ہر علامت (جیسے مستطیل، ڈائمنڈ اور بیضوی) کسی عمل یا فیصلے کو ظاہر کرتی ہے اور تیر مختلف مراحل کے درمیان تعلق اور سمت کی نشاندہی کرتے ہیں۔

(iii) فلوچارٹ ایک بصری انداز میں عمل کو واضح کرتا ہے، جو مجموعی بہاؤ اور ساخت کو سمجھنے کے لیے زیادہ مؤثر ہو سکتا ہے۔

(iv) یہ ایک نظر میں الگورتھم میں شامل مراحل اور فیصلوں کی فوری شناخت کرنے کے لیے مفید ہے۔

مثال: الگورتھم 3 صارفین کے درست نام اور پاس ورڈ کی جانچ پڑتال کے لیے سوڈوکوڈ پیش کرتا ہے۔

```
1:  ProcedureCheckCredentials(username,password)
2:  Input: usernamse,password
3:  Output:Validity message
4:  Begin
5:  validUsername = "user123" {Replace with the actual valid username}
6:  validPassword = "pass123" {Replace with the actual valid password}
7:  if (username = = validUsername) then
8:                  if(password = = validPassword) then
9:                              print"Loginsuccessful"
```

```
10:                 else
11:                                 print"Invalid password"
12:         end if
13:         else
14:                 print "Invalid username"
15:         end if
16:         End
```

سوال 18: الگورتھم کی ٹائم کمپلیکسٹی کیا ہوتی ہے اور یہ ان پٹ کے سائز میں اضافے کے ساتھ کیسے تبدیل ہوتی ہے؟

جواب:

الگورتھم کی سرگرمیاں

الگورتھم کی کارکردگی کا تجزیہ ضروری ہوتا ہے تاکہ معلوم ہو سکے کہ وہ مسائل کو کتنے مؤثر طریقے سے حل کرتے ہیں۔ اس سیکشن میں، ہم الگورتھم کے ڈیزائن اور تشخیص کی تکنیکوں کا جائزہ لیں گے، خاص طور پر ان کی ٹائم اور اسپیس کمپلیکسٹی پر توجہ دیں گے۔

ڈیزائن اور تشخیص کی تکنیک

الگورتھم کا تجزیہ یہ جاننے کے لیے کیا جاتا ہے کہ وہ کتنے مؤثر اور تیز ہیں۔ اس عمل میں الگورتھم کے چلنے کے وقت اور درکار میموری کا جائزہ لیا جاتا ہے تاکہ بہترین طریقہ کار کا انتخاب کیا جا سکے۔

ٹائم کمپلیکسٹی

ٹائم کمپلیکسٹی کسی الگورتھم کی کارکردگی کی پیمائش کا ایک طریقہ ہے، جو یہ بتاتا ہے کہ ان پٹ کے سائز میں اضافے کے ساتھ الگورتھم کے چلنے کا وقت کیسے تبدیل ہوتا ہے۔

مثال: فرض کریں کہ آپ کے پاس ناموں کی ایک فہرست ہے اور آپ کو ایک مخصوص نام تلاش کرنا ہے۔

(i) اگر فہرست میں 10 نام ہوں تو تلاش میں چند سیکنڈ لگ سکتے ہیں۔

(ii) اگر فہرست میں 100 نام ہوں تو تلاش میں مزید وقت لگے گا۔

(iii) اگر فہرست میں 1,000 نام ہوں تو وقت اور بھی زیادہ ہو سکتا ہے۔

جیسے جیسے فہرست لمبی ہوتی جائے گی، تلاش کرنے میں لگنے والا وقت بھی بڑھتا جائے گا۔ ٹائم کمپلیکسٹی کا تجزیہ ہمیں اس اضافے کو سمجھنے میں مدد دیتا ہے۔

کیا آپ جانتے ہیں؟
ٹائم کمپلیکسٹی عموماً Big O نوٹیشن کے ذریعے ظاہر کی جاتی ہے، جیسے O(n)، O(logn) یا O(n2)۔ یہ ہمیں مختلف الگورتھمز کا موازنہ کرنے میں مدد دیتی ہے تاکہ ہم دیکھ سکیں کہ کون سا تیز ہے!

دلچسپ معلومات:
جب آپ ایک الگورتھم لکھ رہے ہوتے ہیں تو اس بات کو مدنظر رکھیں کہ وہ ٹاسک کتنے مراحل میں مکمل کرتا ہے۔ کم مراحل سے مراد تیز الگورتھم ہے۔

سرگرمی
ایک آسان کام کے بارے میں سوچیں، جیسے کسی فہرست میں سب سے بڑے عدد کو تلاش کرنا۔ ایسا کرنے کے لیے آپ جو اقدامات کریں گے اسے لکھیں۔ اب تصور کریں کہ فہرست میں 10 نمبر ہیں، پھر 100 نمبر ہیں۔ اقدامات کیسے بدلتے ہیں؟

کیا آپ جانتے ہیں؟
کچھ الگورتھم ایک ہی کام کو دوسروں کے مقابلے میں بہت تیزی سے انجام دے سکتے ہیں۔ مثال کے طور پر، 100 اشیاء کی فہرست کو ترتیب دینے میں ایک الگورتھم 1 سیکنڈ جب کہ دوسرا الگورتھم 10 سیکنڈ کا وقت لے سکتا ہے!

سوال 19: سپیس کمپلیکسٹی کیا ہے اور اس کے تجزیے کے لیے کن عوامل اور تکنیکوں پر غور کیا جاتا ہے؟

جواب:

سپیس کمپلیکسٹی

سپیس کمپلیکسٹی کسی الگورتھم کی وہ خصوصیت ہے جو ان پٹ سائز کے لحاظ سے استعمال ہونے والی میموری کی مقدار کی پیمائش کرتی ہے۔

میموری کے تقاضے

سپیس کمپلیکسٹی کا تجزیہ کرتے وقت، دو اہم عوامل پر غور کیا جاتا ہے:

ان پٹ کے لیے درکار میموری– دی گئے ڈیٹا کو محفوظ کرنے کے لیے ضروری میموری۔

اضافی میموری– الگورتھم کے عمل کے دوران استعمال ہونے والی عارضی میموری جیسے ویری ایبلز، ارے اور اسٹیکس وغیرہ۔

تشخیص کی تکنیکیں

الگورتھم کی سپیس کمپلیکسٹی جانچنے کے لیے مختلف تکنیکیں استعمال کی جاتی ہیں، جن میں شامل ہیں:

ڈرائی رن– الگورتھم کو بغیر کسی اصل عمل درآمد کے کاغذ پر جانچنا۔

سیمولیشن– کمپیوٹر پر الگورتھم کو چلا کر اس کے میموری کے استعمال کا مشاہدہ کرنا۔

یہ تکنیکیں اس بات کو یقینی بنانے میں مدد دیتی ہیں کہ الگورتھم نہ صرف درست نتائج فراہم کرے بلکہ مؤثر طریقے سے میموری بھی استعمال کرے۔

سوال 20: ڈرائی رن کیا ہوتا ہے اور فلوچارٹ کے ڈرائی رن سے کس طرح منطقی غلطیوں کی نشاندہی کی جاتی ہے؟

جواب:

ڈرائی رن (Dry Run)

ڈرائی رن ایک ایسی تکنیک ہے جس میں الگورتھم کو دستی طور پر نمونے کے ڈیٹا کے ساتھ چلایا جاتا ہے تاکہ کسی بھی ممکنہ غلطیوں کی نشاندہی کی جا سکے۔

فلوچارٹ کا ڈرائی رن

فلوچارٹ کے ڈرائی رن میں، فلوچارٹ کے ہر قدم کو دستی طور پر سمجھا جاتا ہے تاکہ کمپیوٹر کے بغیر الگورتھم کے کام کرنے کے طریقے کو جانچا جا سکے۔ اس عمل سے منطقی غلطیوں کی نشاندہی کرنے اور کنٹرول کے بہاؤ کو بہتر طور پر سمجھنے میں مدد ملتی ہے۔

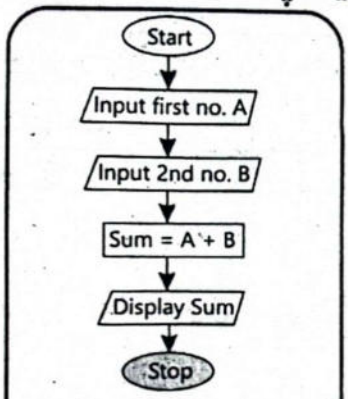
مثال: دو نمبروں کی جمع

دو نمبروں کو جمع کرنے کے لیے درج ذیل فلوچارٹ پر غور کریں:

فلوچارٹ کے ڈرائی رن کے اقدامات:

ابتدائی مرحلے میں، دو نمبروں کو بطور ان پٹ لیا جاتا ہے۔

ان نمبروں کو آپس میں جمع کیا جاتا ہے۔
نتیجہ کو آؤٹ پٹ کے طور پر ظاہر کیا جاتا ہے۔
عمل مکمل ہو جاتا ہے۔
یہ طریقہ فلو چارٹ کی درستگی کو یقینی بنانے اور منطقی مسائل کو دور کرنے میں مدد دیتا ہے۔



شکل 7.7 دو نمبروں کو جمع کرنے کا فلو چارٹ

1. Start
2. Input the first number(e.g.3)

اقدامات:

1.Start
2. Input the first number(e.g.3)
3. Input the second number(e.g.5)
4. Add the two numbers (3+5=8)
5. Output the result(8)
6. Stop

سرگرمی
ڈرائی رن کا ایک فلو چارٹ دو نمبروں میں سے بڑا نمبر معلوم کرنے کے لیے ایک فلو چارٹ بنائیں۔ نمبر 7 اور 4 کے لیے ڈرائی رن کریں۔ ہر قدم اور متغیرات کی قیمتیں (Values of variables) لکھیں۔

سوال 21: سوڈوکوڈ کے ڈرائی رن کا مقصد کیا ہے اور اس کے بنیادی مراحل کون سے ہیں؟

جواب:

سوڈوکوڈ کا ڈرائی رن

سوڈوکوڈ کے ڈرائی رن کا مطلب اسے لائن بہ لائن دستی طور پر چلا کر اس کے عمل کو سمجھنا ہے۔ یہ طریقہ الگورتھم کی منطق اور درستگی کی تصدیق کرنے میں مدد دیتا ہے۔

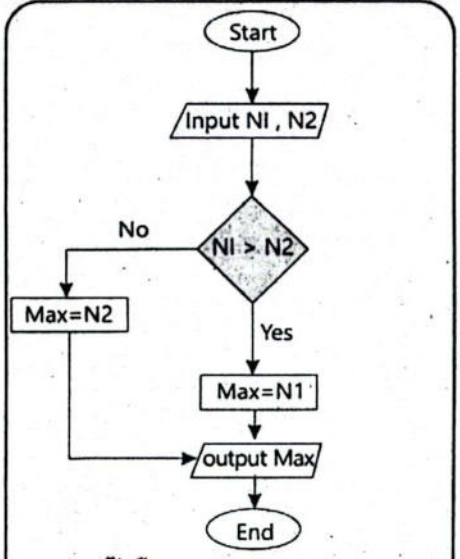
سوڈوکوڈ کے ڈرائی رن کے مراحل

ان پٹ کا تجزیہ – سوڈوکوڈ میں دیے گئے ان پٹ ویلیوز کو منتخب کیا جاتا ہے۔
ہر لائن کا معائنہ – ہر لائن کو ترتیب وار پڑھا اور اس پر عمل درآمد کیا جاتا ہے۔
متغیرات کی جانچ – ہر مرحلے پر متغیرات کی موجودہ قدریں لکھی جاتی ہیں۔
حتمی نتیجہ کا جائزہ – آخری نتیجے کا موازنہ متوقع نتائج سے کیا جاتا ہے تاکہ درستگی کی تصدیق ہو سکے۔
یہ عمل کسی بھی منطقی غلطی کو تلاش کرنے اور الگورتھم کے بہاؤ کو بہتر طور پر سمجھنے میں مدد فراہم کرتا ہے۔

اس سوڈوکوڈ کو ڈرائی رن کرنے کے مراحل:

Algorithm4 Find Max

1. Input:num1,num2
2. if num1>num2then
3. max=num1
4. else
5. max = num 2
6. end if
7. Output: max



شکل 7.8: سب سے بڑا نمبر ڈھونڈنے کا الگورتھم

کیا آپ جانتے ہیں؟
اپنے کوڈ یا الگورتھم کی ڈرائی رننگ کرنے سے ڈویلپمنٹ کے عمل میں غلطیوں کو جلد پکڑنے میں مدد ملتی ہے، جس سے وقت اور محنت کی بچت ہوتی ہے۔ بہت سے پیشہ ور پروگرامرز اور کمپیوٹر سائنسدان اپنے الگورتھم کو صحیح طریقے سے کام کرنے کو یقینی بنانے کے لیے ڈرائی رننگ کو ڈی بگنگ تکنیک کے طور پر استعمال کرتے ہیں!

سوال 22: سیمولیشن کا استعمال الگورتھمز کی جانچ اور ممکنہ حالات کے تجزیے کے لیے کیسے کیا جاتا ہے؟

جواب: سیمولیشن (Simulation)

سیمولیشن ایک ایسا طریقہ ہے جس میں کمپیوٹر پروگراموں کو حقیقی دنیا کے عمل یا نظام کا ماڈل بنانے کے لیے استعمال کیا جاتا ہے۔ یہ طریقہ ہمیں حقیقی زندگی میں تجربات کیے بغیر مختلف الگورتھمز کی جانچ اور ان کے کام کرنے کے طریقے کو سمجھنے میں مدد دیتا ہے۔

سیمولیشن استعمال کرنے کی وجوہات

1۔ ٹیسٹنگ الگورتھم

سیمولیشن کا استعمال الگورتھم کی جانچ کے لیے کیا جا سکتا ہے تا کہ یہ معلوم ہو کہ وہ مختلف قسم کے ڈیٹا کے ساتھ کتنی مؤثر کارکردگی دکھاتا ہے۔

مثال: اگر ہم نمبروں کو ترتیب دینے کا ایک نیا طریقہ آزمانا چاہتے ہیں، تو ہم اسے مختلف ڈیٹا سیٹس پر چلا کر اس کی رفتار اور درستگی کا تجزیہ کر سکتے ہیں۔

2۔ ممکنہ حالات کا تجزیہ

سیمولیشن مختلف حالات تخلیق کرنے میں مدد دیتی ہے تا کہ یہ جانچا جا سکے کہ ان میں ہونے والی تبدیلیوں کے کیا اثرات ہوں گے۔

مثال: اگر ہم پودوں کی نشوونما پر تحقیق کر رہے ہیں، تو ہم مختلف مقدار میں پانی اور سورج کی روشنی کی سیمولیشن کر سکتے ہیں تا کہ معلوم ہو کہ کون سے عوامل پودوں کی نشوونما میں بہترین کردار ادا کرتے ہیں۔

سوال 23: سیمولیشن کے کیا فوائد ہیں اور یہ موسم کی پیش گوئی اور ٹریفک کے بہاؤ میں کیسے مددگار ثابت ہوتی ہے؟

جواب: **سیمولیشن کے فوائد**

کم لاگت

حقیقی تجربات کرنے کے مقابلے میں سیمولیشن چلانا عام طور پر سستا اور تیز تر ہوتا ہے۔

محفوظ

خطرناک حالات کی جانچ کے دوران کسی کو بھی خطرے میں ڈالے بغیر تحقیق کی جاسکتی ہے، جیسے کہ عمارت میں آگ لگنے کی صورت میں حفاظتی اقدامات کا تجزیہ۔

دہرانے کے قابل

ایک ہی سیمولیشن کو مختلف ترتیب کے ساتھ کئی بار چلایا جاسکتا ہے تاکہ یہ دیکھا جاسکے کہ حالات کس طرح تبدیل ہوتے ہیں اور الگورتھم مختلف حالات میں کیسے کارکردگی دکھاتا ہے۔

سیمولیشن کی مثالیں

موسم کی پیش گوئی

ماہرینِ موسمیات موسم کی پیش گوئی کرنے کے لیے سیمولیشن کا استعمال کرتے ہیں۔ وہ درجہ حرارت، نمی اور ہوا کی رفتار جیسے ڈیٹا کو کمپیوٹر ماڈل میں ان پٹ کرتے ہیں تاکہ اندازہ لگایا جاسکے کہ آئندہ چند دنوں میں موسم کیسا ہوگا۔

ٹریفک کا بہاؤ

شہر کے منصوبہ ساز ٹریفک کی سیمولیشن کرتے ہیں تاکہ یہ دیکھا جاسکے کہ سڑکوں یا ٹریفک لائٹس میں تبدیلیاں گاڑیوں کے بہاؤ کو کس طرح متاثر کرسکتی ہیں۔ اس سے سڑکوں کے بہتر ڈیزائن اور ٹریفک جام کو کم کرنے میں مدد ملتی ہے۔

سوال 24: LARP کیا ہے اور یہ الگورتھمز کو سمجھنے میں کیسے مدد دیتا ہے؟

جواب: **لاجک آف الگورتھمز فار ریزولیوشن آف پرابلمز (LARP) کا تعارف**

LARP کا مطلب مسئلے کے حل کے لیے الگورتھم کی منطق ہے۔ یہ ایک دلچسپ اور انٹرایکٹو طریقہ ہے جس کے ذریعے الگورتھم کو عملی طور پر چلا کر ان کے نتائج دیکھے جاسکتے ہیں۔ اسے ایک کھیل کے میدان کے طور پر تصور کریں، جہاں آپ مختلف الگورتھمز کے ساتھ تجربہ کر سکتے ہیں اور سمجھ سکتے ہیں کہ وہ ڈیٹا پر کس طرح عمل کرتے ہیں۔

LARP کیوں اہم ہے؟

الگورتھم کے کام کرنے کے طریقے کو سمجھنا

LARP سیکھنے کا ایک بہترین ذریعہ ہے، کیونکہ یہ ہمیں الگورتھم کے کام کرنے کے طریقے کو گہرائی سے سمجھنے میں مدد دیتا ہے۔ مثال کے طور پر، تصویر 7.9 ایک ایسا الگورتھم ظاہر کرتی ہے جو کسی شخص کی سالانہ تنخواہ پر ٹیکس کے اطلاق کا تعین کرتا ہے۔

مختلف ان پٹ کے آؤٹ پٹ پر اثرات کا تجزیہ

LARP ہمیں یہ جاننے میں مدد دیتا ہے کہ مختلف ان پٹ ویلیوز کا آؤٹ پٹ پر کیا اثر پڑتا ہے۔ اس سے ہم یہ سیکھ سکتے ہیں کہ الگورتھم

مختلف حالات میں کیسے کام کرتا ہے۔

الگورتھم لکھنے اور بہتر بنانے کی مشق

LARP کے ذریعے ہم اپنے الگورتھمز کو خود لکھنے اور ان میں بہتری لانے کی مشق کر سکتے ہیں، جس سے مسئلہ حل کرنے کی مہارتوں میں اضافہ ہوتا ہے۔

سوال 25: LARP کا استعمال کرتے ہوئے الگورتھم کیسے لکھا جاتا ہے اور اس کی ساخت کیا ہوتی ہے؟

جواب:

الگورتھم لکھنا

LARP کا استعمال کرتے ہوئے الگورتھم لکھنا آسان ہوتا ہے، جس کے ذریعے مسائل کے لیے منطقی حل تیار کیے جا سکتے ہیں۔ LARP ایک واضح ٹیکسٹ استعمال کرتا ہے جو "Start" کمانڈ سے شروع ہوتا ہے اور "End" کمانڈ کے ساتھ ختم ہوتا ہے، تاکہ الگورتھم کے ہر مرحلے کو آسانی سے سمجھا جا سکے۔

LARP میں الگورتھم کی ساخت

اس فریم ورک کے اندر، ہدایات کو سادہ انداز میں فراہم کیا جاتا ہے، جیسے:

(i) پیغامات ظاہر کرنے کے لیے write کا استعمال

(ii) ان پٹ حاصل کرنے کے لیے read کا استعمال

(iii) فیصلہ سازی کے لیے if-then-else جیسی اسٹیٹمنٹس

LARP پیچیدہ مسائل کو آسان اور قابلِ فہم مراحل میں تقسیم کر کے سیکھنے والوں کو پیچیدہ کوڈنگ سے بچاتے ہوئے الگورتھم کے منطقی بہاؤ پر توجہ مرکوز کرنے کی سہولت فراہم کرتا ہے۔

LARP کی اہمیت

یہ طریقہ الگورتھم ڈیزائن کے بنیادی تصورات کو سمجھنے میں مدد دیتا ہے اور ساتھ ہی ساتھ واضح اور منطقی سوچ کی حوصلہ افزائی کر کے مسئلہ حل کرنے کی مہارت کو بھی بڑھاتا ہے۔

مثال:

جفت یا طاق نمبر چیک کرنے کا الگورتھم

یہاں ایک سادہ الگورتھم کی مثال دی گئی ہے جو چیک کرتا ہے کہ آیا دیا گیا نمبر جفت ہے یا طاق:

1- Start

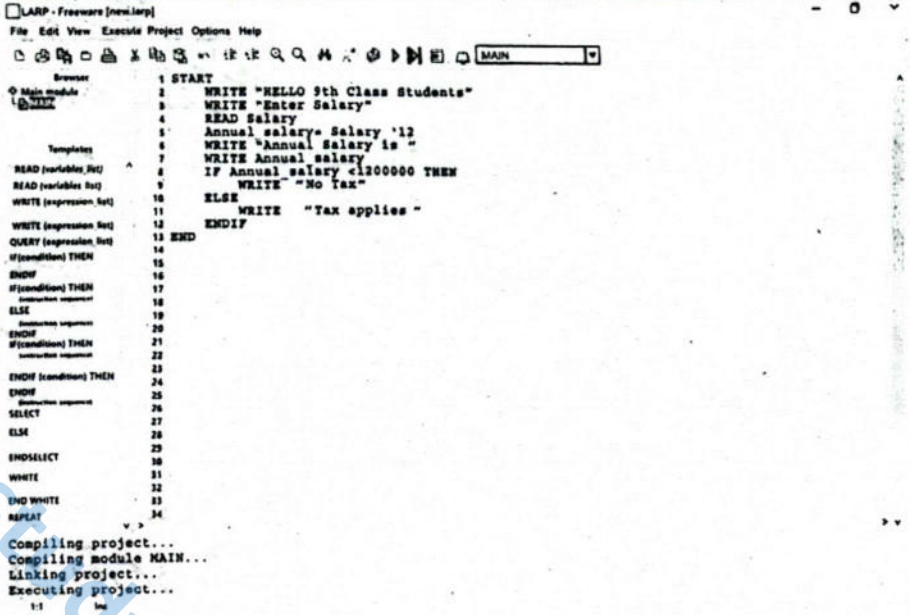
2- ایک نمبر read کریں

3- اگر نمبر 2 سے مکمل تقسیم ہو جائے تو "یہ جفت ہے" لکھیں

4- ورنہ "یہ طاق ہے" لکھیں

5- End

یہ بنیادی مثال LARP کے ذریعے الگورتھم ڈیزائن کے اصولوں کو سیکھنے میں مدد فراہم کرتی ہے۔



سوال 26: LARP میں فلوچارٹ بنانے کا مقصد کیا ہے اور اس میں کون سی بنیادی علامتیں استعمال کی جاتی ہیں؟

جواب: LARP میں فلوچارٹ بنانا

LARP میں فلوچارٹ بنانا الگورتھم کے مراحل کو بصری طور پر پیش کرنے کا عمل ہے۔ اس میں مخصوص فلوچارٹ علامتوں کا استعمال کیا جاتا ہے، جیسے:

(i) مستطیل (Rectangle) → پروسیسنگ کے لیے

(ii) ڈائمنڈ (Diamond) → فیصلے (Decision Making) کے لیے

(ii) پیرالیلوگرام (Parallelogram) → ان پٹ/آؤٹ پٹ آپریشنز کے لیے

LARP میں فلوچارٹ کا اطلاق

فلوچارٹ بنانے کے بعد، اسے LARP کے ٹیکسٹ میں ترجمہ کر کے چلایا جاتا ہے۔ اس عمل میں درج ذیل احکامات شامل ہوتے ہیں:

1- Start (آغاز) 2- Read (ان پٹ لینا) 3- Write (نتائج ظاہر کرنا)

4- if اور else if (فیصلہ سازی کے لیے)

فلوچارٹ کا عملی استعمال

یہ طریقہ طالب علموں کو اپنے الگورتھم کی منطق کو بہتر طور پر تصور کرنے اور اس کے قدم بہ قدم عملدرآمد کو دیکھنے میں مدد دیتا ہے۔

مثال: شکل 7.10 میں ایک فلوچارٹ دکھایا گیا ہے جو یہ طے کرتا ہے کہ آیا کسی طالب علم کا گریڈ "A" سے اوپر ہے یا نہیں۔ فلوچارٹ کو عملی طور پر چلا کر اس کی درستگی کی تصدیق کی جا سکتی ہے۔

یہ دستی نقطہ نظر فلوچارٹ کے کام کرنے کے اصولوں کو سمجھنے اور ان کے مؤثر اطلاق میں مدد فراہم کرتا ہے۔

سوال 27: LARP میں الگورتھم کی غلطیوں کو کیسے شناخت کیا جاتا ہے اور ڈیبگنگ کے کون سے طریقے استعمال کیے جاتے ہیں؟

جواب: غلطی کی شناخت اور ڈیبگنگ

جب ہم LARP میں الگورتھم لکھتے ہیں یا فلوچارٹ بناتے ہیں تو بعض اوقات غلطیاں کر بیٹھتے ہیں، جنھیں غلطیاں یا بگز (Bugs) کہا

جاتا ہے۔ یہ غلطیاں الگورتھم کو صحیح طریقے سے کام کرنے سے روک سکتی ہیں اور متوقع نتائج پر اثر ڈال سکتی ہیں۔

غلطی کی شناخت

غلطیوں کی شناخت ایک ضروری مرحلہ ہے جس میں یہ دیکھا جاتا ہے کہ الگورتھم کہاں غلطی کر رہا ہے۔ اس کے لیے درج ذیل طریقے استعمال کیے جا سکتے ہیں:

(i) آؤٹ پٹ کی جانچ: غلط آؤٹ پٹ آنے پر کوڈ کا بغور جائزہ لیا جاتا ہے۔

(ii) لائن بہ لائن تجزیہ: الگورتھم کو مرحلہ وار چلا کر اس میں موجود کسی منطقی یا تحریری غلطی کا پتا لگایا جاتا ہے۔

ڈیبگنگ (Debugging)

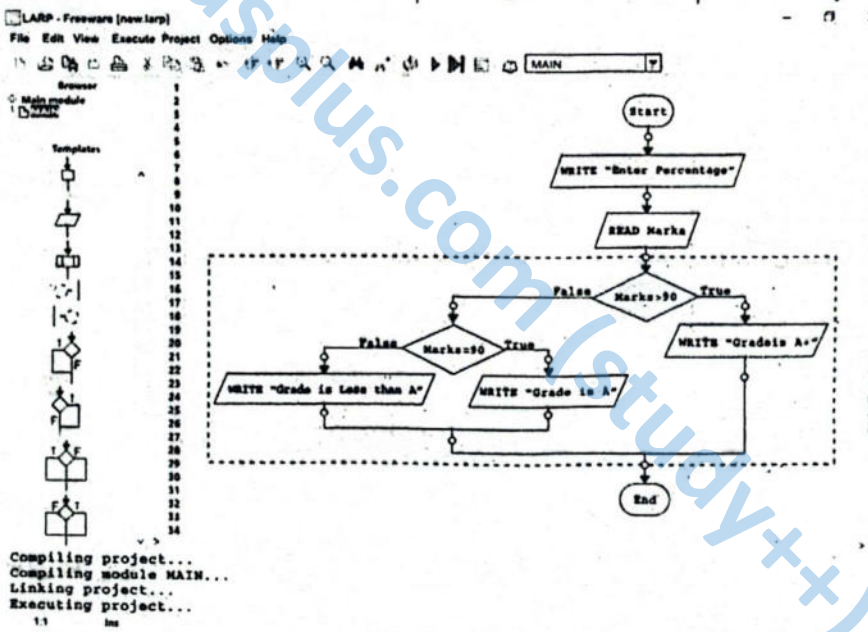
ڈیبگنگ وہ عمل ہے جس کے ذریعے ہم الگورتھم یا فلو چارٹ میں موجود غلطیوں کو تلاش کر کے انھیں درست کرتے ہیں۔ یہ عمل درج ذیل طریقوں سے کیا جا سکتا ہے:

(i) ڈرائی رن: الگورتھم کو کاغذ پر چلانا تاکہ کسی بھی منطقی غلطی کی نشاندہی کی جا سکے۔

(ii) ٹیسٹ کیسز: مختلف ان پٹ ویلیوز دے کر نتائج کا موازنہ کرنا۔

(iii) ترتیب وار چیکنگ: ہر لائن یا مرحلے کو الگ الگ چیک کر کے غلطی کی نشاندہی کرنا۔

یہ عمل یقینی بناتا ہے کہ ہمارا الگورتھم مؤثر اور درست طریقے سے کام کرے۔



سوال 28: الگورتھمز اور فلو چارٹس میں پائی جانے والی تین بنیادی غلطیوں کی وضاحت کریں۔

جواب: **غلطیوں کی اقسام**

الگورتھمز اور فلو چارٹس میں تین بنیادی قسم کی غلطیاں پائی جا سکتی ہیں:

1۔ سِنٹیکس (Syntax) کی غلطیاں

یہ غلطیاں اس وقت ہوتی ہیں جب الگورتھم یا فلو چارٹ میں کوئی تحریری یا ساختی غلطی ہو۔ مثال کے طور پر، کسی ضروری اسٹیپ کو چھوڑ دینا یا

غلط علامت کا استعمال کرنا۔

2۔ رن ٹائم (Run-time) غلطیاں

یہ غلطیاں اس وقت پیدا ہوتی ہیں جب الگورتھم پر عمل درآمد ہو رہا ہو۔ مثال کے طور پر، صفر سے تقسیم کرنے کی کوشش کرنا، جو کہ ایک ناممکن آپریشن ہے اور سسٹم کو کریش کر سکتا ہے۔

کیا آپ جانتے ہیں؟
سنٹیکس کی غلطیوں کو تلاش کرنا سب سے آسان ہے کیونکہ LARP ٹول عام طور پر ان کی نشاندہی کرتا ہے۔ تاہم منطقی غلطیوں کو تلاش کرنا سب سے مشکل ہے کیونکہ الگورتھم اب بھی چلتا ہے لیکن صحیح جوابات نہیں دیتا۔

3۔ منطقی (Logical) غلطیاں

یہ وہ غلطیاں ہیں جو الگورتھم کی منطق میں خرابی کی وجہ سے پیدا ہوتی ہیں۔ مثال کے طور پر، فیصلے کے مرحلے پر غلط سمبلز یا کنڈیشنز کا استعمال کرنا، جس سے الگورتھم غلط نتائج فراہم کرتا ہے۔ یہ غلطیاں الگورتھم کی کارکردگی پر براہِ راست اثر ڈالتی ہیں اور انھیں ڈیبگنگ کے ذریعے درست کرنا ضروری ہوتا ہے۔

سوال 29: ڈیبگنگ کی عام تکنیکیں کون سی ہیں اور یہ الگورتھم کی درستگی میں کیسے مدد دیتی ہیں؟

جواب: ڈیبگنگ کی تکنیکیں

ڈیبگنگ الگورتھم یا فلوچارٹ میں موجود غلطیوں کو تلاش کرنے اور درست کرنے کا عمل ہے۔ یہاں کچھ عام ڈیبگنگ تکنیکیں درج کی جا رہی ہیں:

1۔ مراحل کا جائزہ لیں

اپنے الگورتھم یا فلوچارٹ کے ہر مرحلے کو بغور دیکھیں اور یہ جانچیں کہ کہاں غلطی ہو رہی ہے۔ اس طریقے سے آپ غلطی کی نشاندہی آسانی سے کر سکتے ہیں۔

2۔ کمنٹس کا استعمال

دلچسپ معلومات:
ہمیشہ غلطی کے پیغام کو غور سے پڑھیں یہ عموماً آپ کو بتا دیتے ہیں کہ مسئلہ کہاں پر ہے۔

اپنے الگورتھم میں کمنٹس یا نوٹس شامل کریں تاکہ ہر مرحلے کے مقصد کو واضح کیا جا سکے۔ اس سے آپ کو اور دوسروں کو سمجھنے میں آسانی ہوگی اور غلطیوں کی نشاندہی تیز ہو جائے گی۔

3۔ شرائط کی جانچ

فیصلہ سازی (Decision Making) کے تمام مراحل میں شرائط کو چیک کریں کہ وہ درست طریقے سے لکھی گئی ہیں اور متوقع نتائج فراہم کر رہی ہیں یا نہیں۔

4۔ مسئلے کو آسان بنائیں

الگورتھم کو چھوٹے حصوں میں تقسیم کریں اور ہر حصے کو الگ الگ ٹیسٹ کریں۔ اس طریقہ سے پیچیدہ مسائل کو حل کرنا آسان ہو جاتا ہے اور غلطی کو جلدی پکڑا جا سکتا ہے۔ یہ تکنیکیں ڈیبگنگ کے عمل کو مؤثر بناتی ہیں اور الگورتھم کی درستگی کو یقینی بنانے میں مدد دیتی ہیں۔

سوال 30: LARP میں عام غلطی کے پیغامات کون سے ہوتے ہیں اور ان کی وجوہات کیا ہو سکتی ہیں؟

جواب: LARP میں عام غلطی کے پیغامات

LARP میں کام کرتے وقت آپ کو مختلف قسم کے غلطی کے پیغامات کا سامنا ہو سکتا ہے۔ یہاں کچھ عام غلطیوں کی وضاحت کی جا رہی ہے:

1- گمشدہ مرحلہ (Missing Step)

یہ غلطی اس وقت ظاہر ہوتی ہے جب آپ اپنے الگورتھم میں کوئی ضروری قدم شامل کرنا بھول جاتے ہیں۔ اس کے نتیجے میں الگورتھم صحیح طریقے سے کام نہیں کرتا۔

2- غیر متعینہ ویری ایبلز (Undefined Variables)

یہ اس وقت ہوتا ہے جب آپ کسی ایسے ویری ایبل (Variable) کا استعمال کر رہے ہوتے ہیں جسے پہلے ڈیفائن (Define) نہیں کیا گیا۔ اس سے الگورتھم میں خرابی پیدا ہو سکتی ہے۔

3- غلط آپریشن (Invalid Operations)

یہ اس وقت ہوتا ہے جب آپ کوئی ایسا آپریشن کرنے کی کوشش کرتے ہیں جو ممکن نہیں، جیسے صفر سے کسی عدد کو تقسیم کرنا۔ اس قسم کی غلطی الگورتھم کے رکنے یا غیر متوقع نتائج دینے کا باعث بن سکتی ہے۔ یہ عام غلطیاں سمجھنے اور درست کرنے سے LARP میں الگورتھمز کو بہتر اور مؤثر بنایا جا سکتا ہے۔

سرگرمی
LARP میں ایک سادہ فلوچارٹ بنائیں جو تین نمبروں کی اوسط معلوم کرے۔ اپنے فلوچارٹ میں گرامر کی غلطی، رن ٹائم کی غلطی اور منطقی غلطی متعارف کروائیں۔ پھر زیر بحث ڈی بگنگ تکنیک کا استعمال کرتے ہوئے انھیں ٹھیک کرنے کی کوشش کریں۔

کیا آپ جانتے ہیں؟
ڈی بگنگ کی اصطلاح ایک حقیقی بگ یعنی کیڑے (Moth) سے آئی ہے جو ابتدائی کمپیوٹر میں مسائل پیدا کرتا تھا۔ جب اس کیڑے کو ہٹایا گیا تو اس عمل کو "ڈی بگنگ" کہا گیا۔

خلاصہ

1- کمپیوٹیشنل تھنکنگ کیا ہے اور یہ مسائل کو حل کرنے میں کس طرح مدد دیتی ہے؟

جواب: کمپیوٹیشنل تھنکنگ ایک اہم مہارت ہے جو افراد کو ایسے طریقوں کا استعمال کرتے ہوئے پیچیدہ مسائل کو حل کرنے کے قابل بناتی ہے جو کمپیوٹر سائنس میں شامل عمل کی عکاسی کرتے ہیں۔

2- ڈی کمپوزیشن سے مراد کیا ہے اور یہ مسئلہ حل کرنے میں کیسے مددگار ثابت ہوتی ہے؟

جواب: ڈی کمپوزیشن ایک پیچیدہ مسئلے کو چھوٹے، زیادہ منظم حصوں میں توڑنے کا عمل ہے۔

3- پیٹرن کی شناخت کا عمل کیا ہوتا ہے اور یہ کمپیوٹیشنل تھنکنگ میں کیوں اہم ہے؟

جواب: پیٹرن کی شناخت میں مسائل کے درمیان اور ان کے اندر مماثلت یا نمونے تلاش کرنا شامل ہے۔

4- تجرید (ایبسٹریکشن) کیا ہے اور اس کا بنیادی مقصد کیا ہوتا ہے؟

جواب: تجرید میں پیچیدہ مسائل کو چھوٹے اور زیادہ منظم حصوں میں تقسیم کر کے آسان بنانا اور غیر ضروری مسائل کو نظر انداز کرتے ہوئے صرف ضروری تفصیلات پر توجہ مرکوز کرنا شامل ہے۔

5- الگورتھم کسے کہتے ہیں اور یہ مسئلے کے حل میں کیا کردار ادا کرتا ہے؟

جواب: الگورتھم کسی مسئلے کو حل کرنے یا کسی کام کو مکمل کرنے کے لیے ہدایات کا مرحلہ وار مجموعہ ہے۔

6- کمپیوٹیشنل تھنکنگ میں مسئلے کو سمجھنے کی اہمیت کیا ہے؟

جواب: مسئلے کو سمجھنا مسئلے کو حل کرنے میں پہلا اور سب سے اہم مرحلہ ہے، خاص طور پر کمپیوٹیشنل تھنکنگ میں۔

7- کسی بڑے مسئلے کو آسان بنانے کے لیے کون سی تکنیک استعمال کی جاتی ہے؟

جواب: کسی مسئلے کو آسان بنانے میں اسے چھوٹے، زیادہ منظم ذیلی مسائل میں توڑنا شامل ہے۔

8- بہترین حل کا انتخاب کرنے کے لیے کون سے عوامل مدنظر رکھے جاتے ہیں؟

جواب: بہترین حل کا انتخاب کرنے میں مختلف طریقوں کا جائزہ لینا اور سب سے زیادہ مؤثر کا انتخاب کرنا شامل ہے۔

9- فلوچارٹ کیا ہیں اور یہ کسی عمل کو کیسے ظاہر کرتا ہیں؟

جواب: فلوچارٹ کسی عمل یا نظام میں مراحل کی بصری نمائندگی ہیں، جو تیروں سے منسلک مختلف علامتوں کا استعمال کرتے ہوئے بنائے جاتے ہیں۔

10- سوڈو کوڈ کیا ہوتا ہے اور اسے الگورتھم کی منصوبہ بندی میں کیوں استعمال کیا جاتا ہے؟

جواب: سوڈو کوڈ سادہ اور غیر رسمی زبان کا استعمال کرتے ہوئے الگورتھم کی نمائندگی کرنے کا ایک طریقہ ہے جسے سمجھنا آسان ہے۔ یہ پروگرامنگ زبانوں کی ساخت کو سادہ انگریزی زبان پڑھنے کی قابلیت کے ساتھ جوڑتا ہے، جس سے یہ الگورتھم کی منصوبہ بندی اور وضاحت کے لیے ایک مفید آلہ بن جاتا ہے۔

11- ٹائم کمپلیکسٹی کا کیا مطلب ہے اور یہ الگورتھم کی کارکردگی کو کیسے متاثر کرتی ہے؟

جواب: ٹائم کمپلیسٹی اس بات کی پیائش کرنے کا ایک طریقہ ہے کہ الگورتھم کتنی تیز یا سست کارکردگی کا مظاہرہ کرتا ہے۔ یہ ہمیں بتاتا ہے کہ ان پٹ کے سائز میں اضافے کے ساتھ الگورتھم کا چلنے کا وقت کیسے تبدیل ہوتا ہے۔

12- سپیس کمپلیکسٹی کیا ہے اور یہ کسی الگورتھم کی کارکردگی میں کس طرح اثر انداز ہوتی ہے؟

جواب: سپیس کمپلیکسٹی ان پٹ سائز کے سلسلے میں انگورتم استعمال ہونے والی میموری کی مقدار کی پیائش کرتی ہے۔ ان پٹ کے لیے ضروری میموری اور الگورتھم کے ذریعے استعمال ہونے والی کسی بھی اضافی میموری دونوں پر غور کرنا ضروری ہے۔

13- ڈرائی رن کا کیا مقصد ہے اور یہ کس طرح الگورتھم کی جانچ میں مدد دیتی ہے؟

جواب: ڈرائی رن میں کسی بھی غلطی کی نشاندہی کرنے کے لیے نمونے کے اعداد و شمار کے ساتھ الگورتھم کے ذریعے دستی طور پر جانا شامل ہے۔

14- سیمولیشن کس چیز کو ظاہر کرتی ہے اور اس کا کمپیوٹر سائنس میں کیا استعمال ہے؟

جواب: سیمولیشن اس وقت ہوتا ہے جب ہم حقیقی دنیا کے عمل یا نظام کا ماڈل بنانے کے لیے کمپیوٹر پروگراموں کا استعمال کرتے ہیں۔

15- LARP سے کیا مراد ہے اور یہ سیکھنے کے عمل میں کس طرح مددگار ثابت ہوتا ہے؟

جواب: LARP کا مطلب ہے لائیو الگورتھم چلانا اور مشق کرنا۔ یہ سیکھنے کا ایک دلچسپ اور انٹرایکٹو طریقہ ہے جو سیکھنے والوں کو الگورتھم کو عملی طور پر چلانے اور ان کے نتائج دیکھنے میں مدد دیتا ہے۔

16- ڈی بگنگ کیا ہے اور یہ کمپیوٹیشنل تھنکنگ میں کیوں ضروری ہے؟

جواب: ڈیبگنگ الگورتھم یا فلوچارٹ میں غلطیوں کو تلاش کرنے اور ٹھیک کرنے کا عمل ہے۔

مشق

1۔ درست جواب کا انتخاب کریں:

(i) مندرجہ ذیل میں سے کمپیوٹیشنل تھنکنگ کی بہترین تعریف کون سی ہے؟

(الف) صرف ریاضیاتی حساب کتاب کا استعمال کرتے ہوئے مسائل کو حل کرنے کا ایک طریقہ ہے۔

(ب) ایک مسئلہ حل کرنے کا طریقہ جو منظم، الگورتھمز اور منطقی تھنکنگ کو استعمال کرتا ہے۔

(ج) ایک تکنیک جو خاص طور پر کمپیوٹر پروگرامنگ میں استعمال ہوتی ہے۔

(د) ایک ایسا طریقہ جو حقیقی دنیا کی ایپلی کیشنز کو نظر انداز کرتا ہے۔

(ii) کمپیوٹیشنل تھنکنگ میں مسئلہ کی تقسیم کیوں اہم ہے؟

(الف) یہ مسائل کو چھوٹے، زیادہ منظم حصوں میں تقسیم کر کے آسان بناتا ہے۔

(ب) یہ مزید تفصیلات شامل کر کے مسائل کو پیچیدہ بناتا ہے۔

(ج) یہ مسئلے کو حل کرنے کی ضرورت کو ختم کرتا ہے۔

(د) یہ صرف سادہ مسائل کے لیے مفید ہے۔

(iii) نمونے کی شناخت میں کیا شامل ہیں؟

(الف) مسائل کے اندر مماثلت تلاش کرنا اور ان کا فائدہ اُٹھانا (ب) تکرار کے عناصر کو نظر انداز کرنا

(ج) مسائل کو چھوٹے چھوٹے حصوں میں تقسیم کرنا (د) تفصیلی الگورتھم لکھنا

(iv) کون سی اصطلاح بنیادی خیال پر توجہ مرکوز کرنے کے لیے تفصیلات کو نظر انداز کرنے کے عمل پر مشتمل ہے؟

(الف) ڈی کمپوزیشن (ب) پیٹرن کی شناخت

(ج) تجدید (د) الگورتھم ڈیزائن

(v) مندرجہ ذیل میں سے کون سا کمپیوٹیشنل تھنکنگ کا بنیادی اصول ہے؟

(الف) مسئلے کی تفہیم کو نظر انداز کرنا (ب) مسئلے کو آسان بنانا

(ج) حل کے ڈیزائن سے گریز کرنا (د) بے ترتیب حل کو نافذ کرنا

(vi) الگورتھم کیا ہوتے ہیں:

(الف) ڈیٹا کی فہرستیں (ب) گرافیکل نمائندگی

(ج) کسی مسئلے کو حل کرنے کے لیے مرحلہ وار ہدایات (د) بار بار دہرائے جانے والے پیٹرن

(vii) کمپیوٹیشنل تھنکنگ کے مطابق مسئلہ حل کرنے میں مندرجہ ذیل میں سے کون سا پہلا قدم ہے؟

(الف) حل لکھنا (ب) مسئلے کو سمجھنا

(ج) فلوچارٹ ڈیزائن کرنا (د) حل کا انتخاب کرنا

(viii) فلوچارٹ کس لیے استعمال کیے جاتے ہیں:

(الف) پروگرام کو ڈ کرنے کے لیے (ب) گرافک طور پر الگورتھم کو ظاہر کرنے کے لیے

(ج) ریاضیاتی مساوات کو حل کرنے کے لیے (د) پیٹرنز کی ساخت کے لیے

(ix) سوڈوکوڈ کیا ہے؟

(الف) فلوچارٹ کی ایک قسم

(ب) سادہ زبان کا استعمال کرتے ہوئے الگورتھم کی ایک اعلیٰ سطحی وضاحت

(ج) ایک پروگرامنگ زبان (د) ایک ڈی بگنگ ٹول

(x) فلوچارٹ کو ڈرائی رن کرنے میں کیا شامل ہوتا ہے؟

(الف) پروگرامنگ زبان میں کوڈ لکھنا (ب) نمونہ ڈیٹا کے ساتھ فلوچارٹ کی جانچ

(ج) فلوچارٹ کو سوڈوکوڈ میں تبدیل کرنا (د) فلوچارٹ کی تفصیلات کو نظر انداز کرنا

جوابات

(i)	(ب)	(ii)	(الف)	(iii)	(الف)	(iv)	(ج)	(v)	(ب)
(vi)	(ج)	(vii)	(ب)	(viii)	(ب)	(ix)	(ب)	(x)	(ب)

اہم کثیر الانتخابی سوالات

1۔ کمپیوٹیشنل تھنکنگ کا بنیادی مقصد کیا ہے؟

(الف) کمپیوٹر پروگرامنگ سیکھنا (ب) مسائل کو مؤثر طریقے سے حل کرنا

(ج) کمپیوٹر خریدنے کے طریقے سیکھنا (د) صرف ریاضی کے مسائل حل کرنا

2۔ تقسیم (Decomposition) کا مطلب کیا ہے؟

(الف) کسی بڑے مسئلے کو چھوٹے حصوں میں تقسیم کرنا (ب) کمپیوٹر کو بند کرنا

(ج) کمپیوٹر پروگرام کو چلانا (د) مختلف الگورتھمز کو ایک ساتھ ملانا

3۔ پیٹرن کی شناخت (Pattern Recognition) کا بنیادی فائدہ کیا ہے؟

(الف) مسئلے میں موجود تکرار اور مماثلتوں کو تلاش کرنا (ب) کمپیوٹر پروگرامنگ کے اصول سیکھنا

(ج) صرف ریاضی کے مسائل حل کرنا (د) کمپیوٹر پر گیمز کھیلنا

4۔ تجرید (Abstraction) سے کیا مراد ہے؟

(الف) کسی مسئلے کی غیر ضروری تفصیلات کو نظر انداز کر کے اہم عناصر پر توجہ دینا

(ب) کمپیوٹر کی ہارڈ ویئر اپ گریڈ کرنا (ج) ڈیٹا کو بڑی فائلوں میں جمع کرنا

(د) الگورتھم کو مزید پیچیدہ بنانا

5۔ الگورتھم کی بہترین تعریف کیا ہے؟

(الف) ایک ترتیب شدہ قدم بہ قدم ہدایات کا مجموعہ (ب) کمپیوٹر کی رفتار بڑھانے کا طریقہ

(ج) کمپیوٹر پروگرام کو ڈی بگنگ کرنے کا عمل (د) صرف ریاضی کے فارمولے

6۔ کمپیوٹیشنل تھنکنگ کن شعبوں میں استعمال کی جا سکتی ہے؟

(الف) صرف کمپیوٹر سائنس (ب) صرف انجینئرنگ

(ج) مختلف شعبوں جیسے بایولوجی، معاشیات اور روزمرہ کے مسائل میں (د) صرف ریاضی میں

7۔ اگر ایک طالب علم روزانہ اپنا ہوم ورک بھول جاتا ہے تو پیٹرن کی شناخت کا استعمال کیسے ہو سکتا ہے؟

(الف) ہر روز ہوم ورک کو نظر انداز کرنا (ب) ہوم ورک مکمل کرنے کے لیے ایک یاددہانی ترتیب دینا

(ج) ہوم ورک کے بغیر سکول جانا (د) ہوم ورک کے بارے میں نہ سوچنا

8۔ کون سا آپشن کمپیوٹیشنل تھنکنگ کے اصولوں میں شامل نہیں؟

(الف) تقسیم (ب) پیٹرن کی شناخت

(ج) کوڈنگ (د) تجدید

9۔ درخت لگانے کا درست الگورتھم کون سا ہوگا؟

(الف) درخت لگانے کے لیے پہلے پانی دینا، پھر پودا رکھنا، پھر گڑھا کھودنا

(ب) پہلے جگہ منتخب کرنا، گڑھا کھودنا، پودا لگانا، مٹی ڈالنا، پانی دینا

(ج) پہلے مٹی ڈالنا، پھر پودا نکالنا اور پھر اسے باہر رکھنا (د) پہلے پانی دینا، پھر گڑھا کھودنا اور پودا باہر رکھنا

10۔ کمپیوٹیشنل تھنکنگ کا سب سے پہلا قدم کیا ہوتا ہے؟

(الف) مسئلے کو سمجھنا (ب) فوراً کوڈنگ شروع کرنا

(ج) کمپیوٹر خریدنا (د) کوئی ایک حل چننا

11۔ سوڈوکوڈ کا بنیادی مقصد کیا ہے؟

(الف) پروگرام کو گرافیکل انداز میں ظاہر کرنا (ب) الگورتھم کے مراحل کو عام زبان میں لکھنا

(ج) کوڈ کو براہ راست کسی بھی پروگرامنگ لینگویج میں تبدیل کرنا (د) پروگرام کو کمپائل کرنا

12۔ فلوچارٹ میں مستطیل (Rectangle) کس چیز کی نمائندگی کرتا ہے؟

(الف) فیصلہ (Decision) (ب) ان پٹ/آؤٹ پٹ

(ج) پروسیسنگ (د) اختتام (End)

13۔ سوڈوکوڈ میں کون سا عنصر استعمال نہیں کیا جاتا؟

(الف) سادہ زبان (ب) پروگرامنگ زبان کا مخصوص سنٹیکس

(ج) الگورتھم کے مراحل (د) مشروط جملے (Conditional statements)

14۔ فلوچارٹ میں تیر (Arrow) کا استعمال کیوں کیا جاتا ہے؟

(الف) الگورتھم کے اقدامات کو جوڑنے کے لیے (ب) ڈیٹا کو ذخیرہ کرنے کے لیے

(ج) حساب کتاب کرنے کے لیے (د) پروگرام کو کمپائل کرنے کے لیے

15۔ سوڈوکوڈ کے کون سے فائدے ہیں؟

(الف) یہ گرافیکل ہوتا ہے (ب) یہ عام زبان میں لکھا جاتا ہے اور آسانی سے سمجھا جا سکتا ہے

(ج) اس میں کمپیوٹر کوڈ لکھنے کی ضرورت نہیں ہوتی (د) دونوں ب اور ج

16۔ فلوچارٹ کو عام طور پر کس کے ذریعے تیار کیا جاتا ہے؟

(الف) ورڈ پراسیسر (ب) گرافکس ڈیزائننگ سافٹ ویئر

(ج) ڈایاگرام بنانے کے مخصوص ٹولز (د) ویب براؤزر

17- فلوچارٹ میں بیضوی (Oval) علامت کس چیز کی نمائندگی کرتی ہے؟

(الف) ان پٹ/آؤٹ پٹ (ب) فیصلہ (Decision)

(ج) عمل (Process) (د) الگورتھم کا آغاز اور اختتام

18- سوڈوکوڈ کا استعمال کیوں کیا جاتا ہے؟

(الف) پیچیدہ کوڈنگ سے پہلے الگورتھم کی وضاحت کے لیے (ب) کمپیوٹر پروگرام کے لیے براہ راست کوڈ لکھنے کے لیے

(ج) گرافیکل ریپریزنٹیشن کے لیے (د) الگورتھم کے منطقی غلطیاں تلاش کرنے کے لیے

19- اگر کسی الگورتھم میں زیادہ گرافیکل وضاحت درکار ہو، تو کون سا طریقہ زیادہ مؤثر ہوگا؟

(الف) سوڈوکوڈ (ب) فلوچارٹ

(ج) متن پر مبنی ہدایات (د) کوئی بھی طریقہ مؤثر نہیں

20- سوڈوکوڈ اور فلوچارٹ میں بنیادی فرق کیا ہے؟

(الف) سوڈوکوڈ گرافیکل ہوتا ہے، جبکہ فلوچارٹ تحریری ہوتا ہے

(ب) فلوچارٹ الگورتھم کو بصری انداز میں پیش کرتا ہے، جبکہ سوڈوکوڈ تحریری شکل میں ہوتا ہے

(ج) سوڈوکوڈ کمپیوٹر پروگرامنگ کی زبانوں میں براہ راست ترجمہ کیا جا سکتا ہے، جبکہ فلوچارٹ نہیں

(د) دونوں ب اور ج

جوابات:

-1	(ب)	-2	(الف)	-3	(الف)	-4	(الف)	-5	(الف)
-6	(ج)	-7	(ب)	-8	(ج)	-9	(ب)	-10	(الف)
-11	(ب)	-12	(ج)	-13	(ب)	-14	(الف)	-15	(د)
-16	(ج)	-17	(د)	-18	(الف)	-19	(ب)	-20	(د)

مختصر سوالات

(i) کمپیوٹیشنل تھنکنگ کی وضاحت کریں۔

جواب: کمپیوٹیشنل تھنکنگ ایک مسئلہ حل کرنے کی مہارت ہے جس میں بڑے اور پیچیدہ مسائل کو آسان حصوں میں تقسیم کر کے حل کیا جاتا ہے۔ اس میں تجزیہ، الگورتھمز، پیٹرن کی شناخت اور تجدید جیسے عناصر شامل ہوتے ہیں، جو کمپیوٹر سائنس اور دیگر شعبوں میں کارآمد ہوتے ہیں۔

(ii) کمپیوٹیشنل تھنکنگ میں Decomposition کیا ہے؟

جواب: Decomposition کا مطلب ہے کہ کسی پیچیدہ مسئلے کو چھوٹے، قابلِ حل حصوں میں تقسیم کرنا۔ اس سے مسئلے کو سمجھنا اور اس کے حل تلاش کرنا آسان ہو جاتا ہے، جیسے ایک بڑے سافٹ ویئر کو چھوٹے ماڈیولز میں تقسیم کرنا۔

(iii) Pattern Recognition کو ایک مثال کے ساتھ بیان کریں۔

جواب: Pattern Recognition میں مسئلے کے اندر موجود مشابہتوں اور پیٹرنز کو پہچانا جاتا ہے۔ مثال کے طور پر، کیلکولیشن کے مسائل میں

اگر کوئی ایک جیسا فارمولہ بار بار آ رہا ہو تو اسے شناخت کر کے ایک عام طریقہ بنایا جا سکتا ہے، جیسے ضرب کے جدول کی شناخت۔

(iv) مسئلہ حل کرنے میں تجرید اور اس کی اہمیت کی وضاحت کریں۔

جواب: تجرید (Abstraction) کا مطلب ہے غیر ضروری تفصیلات کو چھوڑ کر مسئلے کے اہم حصے پر توجہ دینا۔ اس سے مسئلہ حل کرنے میں آسانی ہوتی ہے، جیسے کسی نقشے میں صرف سڑکوں اور اہم مقامات کو دکھایا جاتا ہے، غیر ضروری تفصیلات نہیں۔

(v) الگورتھم کیا ہے؟

جواب: الگورتھم ہدایات کا ایک منظم مجموعہ ہوتا ہے جو کسی مسئلے کو مرحلہ وار حل کرنے کے لیے استعمال ہوتا ہے۔ مثال کے طور پر، چائے بنانے کے مراحل کو اگر ترتیب وار لکھا جائے تو یہ ایک الگورتھم ہوگا۔

(vi) مسئلہ کی تفہیم کمپیوٹیشنل تھنکنگ میں کس طرح مدد کرتی ہے؟

جواب: مسئلہ کی صحیح تفہیم سے یہ طے کرنا آسان ہو جاتا ہے کہ کون سی تکنیک استعمال کرنی ہے، جیسے Decomposition, Pattern Recognition یا الگورتھمز۔ اگر مسئلہ اچھی طرح سمجھ میں نہ آئے تو اس کا حل بھی مؤثر نہیں ہوگا۔

(vii) فلوچارٹ کیا ہیں اور وہ کس طرح استعمال ہوتے ہیں؟

جواب: فلوچارٹ کسی عمل یا نظام میں بصری نمائندگی ہیں۔ جو تیروں سے منسلک مختلف علامتوں کا استعمال کرتے ہوئے بنائے جاتے ہیں۔

(viii) سوڈوکوڈ کے مقصد کی وضاحت کریں۔

جواب: سوڈوکوڈ سادہ اور غیر رسمی زبان کا ستعمال کرتے ہوئے الگورتھم کو پیش کرنے کا ایک طریقہ ہے جسے سمجھنا آسان ہے۔ یہ پروگرامنگ کی ساخت کو سادہ انگریزی جیسی پڑھنے میں آسان زبان کا استعمال کرتا ہے، جس سے یہ الگورتھم کی منصوبہ بندی اور وضاحت کے لیے ایک ذریعہ بن جاتا ہے۔

(ix) آپ فلوچارٹ اور سوڈوکوڈ کے درمیان فرق کیسے کرتے ہیں؟

جواب: فلوچارٹ گرافیکل نمائندگی ہے، جبکہ سوڈوکوڈ تحریری شکل میں الگورتھم کو بیان کرتا ہے۔ فلوچارٹ میں مختلف مراحل کو شکلوں کے ذریعے دکھایا جاتا ہے، جبکہ سوڈوکوڈ میں آسان الفاظ میں ترتیب وار ہدایات لکھی جاتی ہیں۔

(x) ڈرائی رن کیا ہے اور یہ کیوں ضروری ہے؟

جواب: ڈرائی رن کسی الگورتھم یا پروگرام کو بغیر کمپیوٹر پر چلائے دستی طور پر جانچنے کا عمل ہے۔ یہ غلطیوں کی نشاندہی اور درستگی کے لیے ضروری ہوتا ہے تاکہ پروگرام کی کارکردگی کو بہتر بنایا جا سکے۔

(xi) LARP کیا ہے اور الگورتھم سیکھنے میں اس کی اہمیت کی وضاحت کریں۔

جواب: LARP (Live Action Role Playing) ایک عملی سرگرمی ہے جس میں لوگ کردار ادا کر کے کسی تصور کو سیکھتے ہیں۔ الگورتھمز سیکھنے میں یہ تکنیک پیچیدہ تصورات کو آسان بنانے میں مدد دیتی ہے، جیسے طلبہ کو کسی الگورتھم کے کردار سونپ کر اس کا عملی مظاہرہ کروایا جائے۔

(xii) دو ڈی بلنگ تکنیکس لکھیں اور ان کی وضاحت کریں۔

جواب: دو ڈی بلنگ تکنیکس میں بریڈتھ فرسٹ سرچ (BFS) اور ڈیپتھ فرسٹ سرچ (DFS) شامل ہیں، جو گراف یا نیٹ ورک میں راستے تلاش کرنے کے لیے استعمال ہوتے ہیں۔ BFS قریبی راستوں کو پہلے چیک کرتا ہے، جبکہ DFS ایک راستے پر گہرائی تک جاتا ہے اور پھر پیچھے آتا ہے۔

اہم مختصر جوابی سوالات

1۔ کمپیوٹیشنل تھنکنگ کیا ہے؟

جواب: کمپیوٹیشنل تھنکنگ ایک مسئلہ حل کرنے کی مہارت ہے جو کمپیوٹر سائنس کے اصولوں پر مبنی ہوتی ہے۔ اس میں بڑے مسائل کو چھوٹے حصوں میں تقسیم کرنا، پیٹرن کی شناخت، تجرید اور الگورتھمز کا استعمال شامل ہوتا ہے۔

2۔ تقسیم (Decomposition) کیوں ضروری ہے؟

جواب: تقسیم ایک پیچیدہ مسئلے کو چھوٹے، قابلِ حل حصوں میں تقسیم کرنے کا عمل ہے۔ اس سے مسئلے کو سمجھنا اور حل تلاش کرنا آسان ہو جاتا ہے، جیسے سکول کی تقریب کی منصوبہ بندی کو مختلف کاموں میں بانٹنا۔

3۔ پیٹرن کی شناخت (Pattern Recognition) کا فائدہ کیا ہے؟

جواب: پیٹرن کی شناخت ہمیں مسائل میں مماثلت تلاش کرنے میں مدد دیتی ہے، جس سے ہم ماضی کے تجربات کی بنیاد پر نئے مسائل کو زیادہ مؤثر طریقے سے حل کر سکتے ہیں، جیسے مربع کے رقبے کے حساب میں ترتیب کا استعمال۔

4۔ تجرید (Abstraction) کا مطلب کیا ہے؟

جواب: تجرید ایک مسئلے کے غیر ضروری عناصر کو ہٹا کر صرف اہم پہلوؤں پر توجہ مرکوز کرنے کا عمل ہے، جیسے چائے بنانے کے بنیادی مراحل کو نمایاں کرنا اور کم ضروری تفصیلات کو نظر انداز کرنا۔

5۔ الگورتھم (Algorithm) کیا ہوتا ہے؟

جواب: الگورتھم کسی مسئلے کو حل کرنے کے لیے ایک مرحلہ وار ہدایات کا مجموعہ ہوتا ہے، جیسے درخت لگانے یا کیک بنانے کے لیے مخصوص مراحل پر عمل کرنا۔

6۔ کمپیوٹیشنل تھنکنگ کیوں ضروری ہے؟

جواب: یہ صرف کمپیوٹر سائنس تک محدود نہیں بلکہ روزمرہ کے مسائل حل کرنے، منصوبہ بندی کرنے اور بہتر فیصلے لینے میں مدد دیتی ہے، جیسے سفر کی منصوبہ بندی یا ہوم ورک کو مؤثر طریقے سے مکمل کرنا۔

7۔ الگورتھم کی روزمرہ زندگی میں مثال دیں؟

جواب: دانت صاف کرنے، سکول کے بیگ کو پیک کرنے، کھانا پکانے اور سڑک پار کرنے کے لیے بھی ہم الگورتھم جیسے مرحلہ وار عمل پر عمل کرتے ہیں۔

8۔ کمپیوٹیشنل تھنکنگ میں مسئلے کی تفہیم کیوں ضروری ہے؟

جواب: کسی بھی مسئلے کا درست حل نکالنے کے لیے پہلے اس کو اچھی طرح سمجھنا ضروری ہے۔ اس میں مسئلے کی تفصیلات جاننا، ضروریات کا تعین کرنا اور ممکنہ رکاوٹوں کا اندازہ لگانا شامل ہوتا ہے۔

9۔ فلوچارٹ اور سوڈوکوڈ میں کیا فرق ہے؟

جواب: فلوچارٹ ایک گرافیکل طریقہ ہے جو مسئلے کو ڈایاگرام کی شکل میں ظاہر کرتا ہے، جبکہ سوڈوکوڈ ایک تحریری طریقہ ہے جو الگورتھم کو سادہ الفاظ میں لکھنے کے لیے استعمال ہوتا ہے۔

10۔ کمپیوٹیشنل تھنکنگ کن شعبوں میں استعمال ہوتی ہے؟

جواب: یہ کمپیوٹر سائنس، انجینئرنگ، میڈیکل، بزنس، تعلیم اور روزمرہ کی زندگی کے مسائل کو حل کرنے میں مدد دیتی ہے، جیسے طبی تشخیص، ڈیٹا تجزیہ

اور لاجسٹک مینجمنٹ۔

11- فلوچارٹ اور سوڈوکوڈ میں بنیادی فرق کیا ہے؟

جواب: فلوچارٹ ایک بصری آلہ ہے جو گرافیکل علامتوں اور تیروں کے ذریعے الگورتھم کو ظاہر کرتا ہے، جبکہ سوڈوکوڈ ایک متنی وضاحت ہے جو انسانی فہم کے لیے آسان زبان میں لکھی جاتی ہے۔

12- سوڈوکوڈ کس طرح الگورتھم کو بہتر طریقے سے بیان کرتا ہے؟

جواب: سوڈوکوڈ سادہ اور ترتیب وار جملوں میں الگورتھم کے مراحل کی وضاحت کرتا ہے، جسے بعد میں کسی بھی پروگرامنگ زبان میں آسانی سے تبدیل کیا جا سکتا ہے۔

13- فلوچارٹ کی کون سی خصوصیت اسے سمجھنے میں آسان بناتی ہے؟

جواب: فلوچارٹ گرافیکل علامتوں جیسے مستطیل (عمل)، ڈائمنڈ (فیصلہ) اور بیضوی (آغاز/اختتام) کے ذریعے الگورتھم کے بہاؤ کو واضح کرتا ہے، جس سے سمجھنا آسان ہو جاتا ہے۔

14- ٹائم کمپلیکسٹی کیوں اہم ہے؟

جواب: ٹائم کمپلیکسٹی الگورتھم کی کارکردگی کو ناپنے کا پیمانہ ہے، جو بتاتا ہے کہ ان پٹ سائز بڑھنے پر اس کا وقت کیسے تبدیل ہوتا ہے۔ یہ الگورتھم کے موازنے میں مدد دیتی ہے۔

15- OBig نوٹیشن کس لیے استعمال کی جاتی ہے؟

جواب: OBig نوٹیشن الگورتھم کی ٹائم کمپلیکسٹی ظاہر کرنے کے لیے استعمال کی جاتی ہے، جیسے O(log n), O(n) یا $O(n^2)$، تاکہ اندازہ لگایا جا سکے کہ وہ کتنا مؤثر ہے۔

16- سپیس کمپلیکسٹی کا مطلب کیا ہے؟

جواب: سپیس کمپلیکسٹی وہ میموری ماپنے کا وہ طریقہ ہے جو الگورتھم اپنے ان پٹ کے مطابق استعمال کرتا ہے، بشمول متغیرات اور عارضی اسٹوریج۔

17- ڈرائی رن کیا ہوتا ہے؟

جواب: ڈرائی رن میں الگورتھم یا فلوچارٹ کو دستی طور پر مختلف ان پٹ کے ساتھ چلا کر اس کی درستگی اور لاجک کی جانچ کی جاتی ہے۔

18- سیمولیشن الگورتھم کی کارکردگی کو کیسے جانچتی ہے؟

جواب: سیمولیشن حقیقی دنیا کے حالات کی نقل تیار کر کے الگورتھم کی مؤثریت اور ممکنہ نتائج کا تجزیہ کرتی ہے، جیسے موسم کی پیش گوئی یا ٹریفک کنٹرول۔

19- LARP کا مقصد کیا ہے؟

جواب: LARP(Logic of Algorithm for Resolution of Problems) الگورتھم کی لاجک کو عملی طور پر سمجھنے اور تجربات کے ذریعے مسائل حل کرنے میں مدد دیتا ہے۔

20- ڈیبگنگ میں کن اہم غلطیوں کو تلاش کیا جاتا ہے؟

جواب: ڈیبگنگ میں تین بنیادی غلطیاں تلاش کی جاتی ہیں:

(i) سنٹیکس غلطیاں (غلط کوڈنگ)

(ii) رن ٹائم غلطیاں (عمل کے دوران پیش آنے والی خرابی)

(iii) منطقی غلطیاں (غلط نتائج پیدا کرنے والی لاجک)

3۔ تفصیلی سوالات:

(i) ایک الگورتھم لکھیں جو کسی طالب علم کے حاصل کردہ نمبروں کی بنیاد پر گریڈ تفویض کرے۔ درج ذیل گریڈنگ سسٹم کا استعمال کریں:

90 اور اس سے زیادہ	:	A+
80 سے 89	:	A
70 سے 79	:	B
60 سے 69	:	C
60 سے نیچے	:	F

جواب: یہاں ایک سادہ الگورتھم دیا گیا ہے جو کسی طالب علم کے نمبروں کی بنیاد پر گریڈ تفویض کرے گا:

الگورتھم برائے گریڈ تفویض

1- آغاز کریں
2- طالب علم کے حاصل کردہ نمبر (marks) ان پٹ کریں۔
3- اگر $marks \geq 90$ ہو تو "A+" گریڈ تفویض کریں۔
4- ورنہ اگر marks 80 سے 89 کے درمیان ہو تو "A" گریڈ تفویض کریں۔
5- ورنہ اگر marks 70 سے 79 کے درمیان ہو تو "B" گریڈ تفویض کریں۔
6- ورنہ اگر marks 60 سے 69 کے درمیان ہو تو "C" گریڈ تفویض کریں۔
7- اگر marks 60 سے کم ہو تو "F" گریڈ تفویض کریں۔
8- گریڈ کو آؤٹ پٹ کریں۔
9- اختتام کریں

سوڈو کوڈ

```
START
        INPUTmarks
        IF marks ≥ 90 THEN
                PRINT "Grade:A+"
        ELSE IF marks ≥ 80THEN
                PRINT"Grade:A"
        ELSE IFmarks ≥ 70THEN
                PRINT"Grade:B"
        ELSE IFmarks ≥ 60THEN
                PRINT"Grade:C"
        ELSE
                PRINT"Grade:F"
        END IF
END
```

Python میں کوڈ

```
marks = int(input(":طالب علم کے نمبر درج کریں"))
if marks>=90:
        print("Grade:A+")
elif marks> = 80:
        print("Grade:A")
elif marks> = 70:
        print("Gracle:B")
```

```
elif marks> = 60:
        print("Grade:C")
else:
        print("Grade:F")
```

یہ کوڈ طالب علم کے نمبروں کو ان پٹ لیتا ہے اور گریڈنگ سسٹم کے مطابق متعلقہ گریڈ تفویض کرتا ہے۔

(ii) وضاحت کریں کہ آپ ایک پیچیدہ کمپیوٹیشنل مسئلے کو حل کرنے کے لیے الگورتھم ڈیزائن کے طریقوں، جیسے فلوچارٹ اور سوڈو کوڈ کا استعمال کیسے کریں گے۔ ایک تفصیلی مثال کے ساتھ اپنی وضاحت کو بیان کریں۔

جواب: دیکھیے سوال نمبر 1

(iii) کمپیوٹیشنل تھنکنگ کی وضاحت کریں اور جدید مسائل کو حل کرنے میں اس کی اہمیت کو بیان کریں۔ مثالیں فراہم کریں کہ مختلف شعبوں میں کمپیوٹیشنل تھنکنگ کو کس طرح لاگو کیا جا سکتا ہے۔

جواب: دیکھیے سوال نمبر 2، 3

(iv) کمپیوٹیشنل تھنکنگ میں تقسیم کے تصور پر تبادلۂ خیال کریں۔ یہ کیوں اہم ہے؟

جواب: دیکھیے سوال نمبر 3

(v) کمپیوٹیشنل تھنکنگ کے حوالے سے پیٹرن کی شناخت کی وضاحت کریں۔ یہ مسئلہ حل کرنے میں کس طرح مدد کرتی ہے؟

جواب: دیکھیے سوال نمبر 1

(vi) کمپیوٹیشنل تھنکنگ میں Abstraction کیا ہے؟ اس کی اہمیت پر تبادلۂ خیال کریں اور اس کی مثالیں فراہم کریں کہ پیچیدہ مسائل کو آسان بنانے کے لیے کس طرح اس کا استعمال کیا جا سکتا ہے۔

جواب: دیکھیے سوال نمبر 8

(vii) الگورتھم کیا ہے اس کی وضاحت کریں اور کمپیوٹیشنل تھنکنگ میں اس کے کردار کی وضاحت کریں۔ کسی مخصوص مسئلے کو حل کرنے کے لیے الگورتھم کی تفصیلی مثال فراہم کریں اور متعلقہ فلوچارٹ بنائیں۔

جواب: دیکھیے سوال نمبر 6

(viii) الگورتھم ڈیزائن کے طریقوں کے طور پر فلوچارٹ اور سوڈو کوڈ کا موازنہ کریں۔ ہر طریقہ کار کے فوائد اور نقصانات پر تبادلۂ خیال کریں اور ایسی مثالیں فراہم کریں جہاں ایک کو دوسرے پر ترجیح دی جا سکتی ہے۔

جواب: دیکھیے سوال نمبر 10

(ix) فلوچارٹ اور سوڈو کوڈ دونوں کے حوالے سے ڈرائی رن کے تصور کی وضاحت کریں۔ ڈرائی رن کرنے سے الگورتھم کے درست ہونے کی تصدیق کرنے میں کس طرح مدد ملتی ہے؟

جواب: دیکھیے سوال نمبر 20

(x) LARP کیا ہے؟ الگورتھم سیکھنے اور مشق کرنے میں اس کی اہمیت پر تبادلۂ خیال کریں۔

جواب: دیکھیے سوال نمبر 24

(xi) LARP کمپیوٹیشنل تھنکنگ کے اصولوں کی تفہیم اور اطلاق کو کس طرح بہتر بناتا ہے؟ ایسی مثال دیں کہ جہاں LARP کو الگورتھم کو بہتر بنانے کے لیے استعمال کیا جا سکتا ہے۔

جواب: دیکھیے سوال نمبر 25