یونٹ 7 کمپیوٹیشنل تھنکنگ

سوال 1: کمپیوٹیشنل تھنکنگ کی وضاحت کریں اور جدید مسائل کو حل کرنے میں اس کی اہمیت کو بیان کریں۔ مثالیں بیان کریں کہ مختلف شعبوں میں کمپیوٹیشنل تھنکنگ کو کس طرح لاگو کیا جا سکتا ہے؟

جواب: نیٹ ورک بطور نظام: کمپیوٹیشنل تھنکنگ (CT) کسی مسئلہ کو حل کرنے کا ایک عمل ہے جس میں پیچیدہ مسائل کو اس طرح حل کرنے کے لیے مہارتوں اور تکنیکوں کا ایک مجموعہ شامل ہے جس پر کمپیوٹر بھی عمل درآمد کر سکتا ہے۔ اس طریقہ کار کو کمپیوٹر سائنس سے باہر بھی مختلف شعبوں میں استعمال کیا جا سکتا ہے، جیسے بیالوجی، ریاضی اور یہاں تک کہ روزمرہ کی زندگی۔

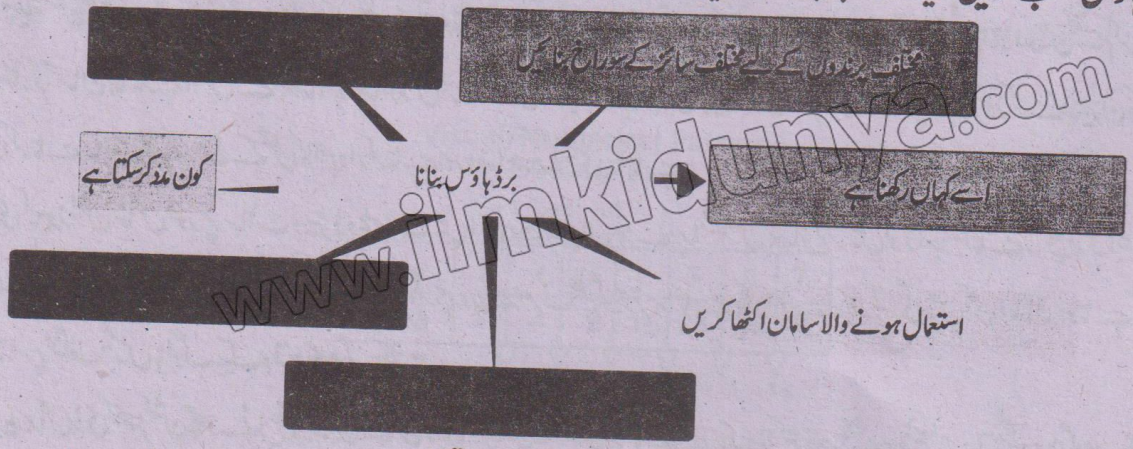
تقسیم یا تجزیہ (Decomposition):

تقسیم ایک پیچیدہ مسئلے کو چھوٹے، زیادہ منظم حصوں میں تقسیم کرنے کا عمل ہے۔ مثال کے طور پر، اسکول کی تقریب کی منصوبہ بندی کرتے وقت، آپ کام کو کچھ حصوں میں تقسیم کر سکتے ہیں جیسے تاریخ کا انتخاب کرنا، مقام کی بکنگ کرنا، دعوت نامے بھیجنا اور سرگرمیوں کا اہتمام کرنا۔

وضاحت:

کمپیوٹیشنل تھنکنگ تقسیم میں ایک اہم قدم ہے اس میں ایک پیچیدہ مسئلے کو چھوٹے، قابل انتظام حصوں میں تقسیم کرنا شامل ہے۔ آیئے پرندوں کے گھر کی تعمیر کی مثال لیتے ہیں۔ یہ کام شروع میں مشکل لگ سکتا ہے لیکن اگر ہم اسے چھوٹے حصوں میں تقسیم کر دیں تو، ایک وقت میں ہم ایک حصے کو سنبھال سکتے ہیں۔ یہاں بتایا گیا ہے کہ ہم پرندوں کے گھر کی تعمیر کے کام کو کس طرح تقسیم کر سکتے ہیں۔ تصویر پرندوں کے گھر کی تعمیر کے لیے تقسیم کو ظاہر کرتی ہے۔

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



شکل 7.1: پرندوں کے گھر کی تقسیم

www.ilmkidunya.com

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 کی ترتیب دینا۔

سرگرمی 1:

- کسی کام کو ڈی کمپوز کریں۔ ایک پیچیدہ کام کے بارے میں سوچیں جو آپ باقاعدگی سے کرتے ہیں، جیسے اسکول 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 تبادلہ خیال کریں کہ کس طرح تقسیم کام کو سنبھالنے میں آسان بناتی ہے۔

جواب: نمونہ: کمپلیکس ٹاسک۔ کھانا پکانا

1۔ **مینو بنانا:** اس بات کی تصدیق کرنا کہ کون سا کھانا تیار کرنے کے لیے اشیا کی فراہمی، متوازن غذا اور کم وقت میں درکار ہوگا۔

2۔ **اجزاء اکٹھا کرنا:** کھانا بنانے کے لیے گھر کا سامان چیک کریں اور اجزاء کو پورا کرنے کے لیے مارکیٹ سے شاپنگ کریں۔

3۔ **اجزاء کو تیار کرنا:** کھانے کی ترکیب کے مطابق اجزا کو چھیلیں، کاٹیں اور تیار کریں۔

4۔ **کھانا پکائیں:** کھانے کی ترکیب کو مدِنظر رکھتے ہوئے کھانے کے اجزاء کو اُبالیں اور پکائیں۔

5۔ **ٹیبل سجائیں:** ٹیبل کو پلیٹس، چمچوں اور دیگر کھانے کی اشیاء سے سجائیں۔

6۔ **کھانا تقسیم کریں:** کھانے کو باؤل اور ڈشز میں ڈال کر تقسیم کریں۔

7۔ **صفائی کرنا:** کھانے کی ٹیبل اور برتن اور فرش کو صاف کریں۔ برتنوں کو ان کی مخصوص جگہ پر رکھ دیں۔

سوال 2: کمپیوٹیشنل تھنکنگ میں تقسیم کے تصور پر تبادلہ خیال کریں۔ یہ کیوں اہم ہے؟

جواب: کمپیوٹیشنل تھنکنگ میں تقسیم کا تصور (Decomposition): تقسیم ایک پیچیدہ مسئلے کو چھوٹے، زیادہ منظم حصوں میں تقسیم کرنے کا عمل ہے۔ مثال کے طور پر، اسکول کی تقریب کی منصوبہ بندی کرتے وقت، آپ کام کو کچھ حصوں میں تقسیم کر سکتے ہیں جیسے تاریخ کا انتخاب کرنا، مقام کی بکنگ کرنا، دعوت نامے بھیجنا اور سرگرمیوں کا اہتمام کرنا۔

اہمیت:

- کمپیوٹیشنل تھنکنگ میں ڈی کمپوزیشن ایک کلیدی تصور ہے جو کسی بڑے مسئلے یا نظام کو چھوٹے، زیادہ قابلِ انتظام اجزاء میں تقسیم کرتا ہے۔ یہ تکنیکی مسئلہ کو آسان بناتا ہے، جسے سمجھنا، جانچنا اور حل کرنا آسان ہو جاتا ہے۔ ڈی کمپوزیشن انفرادی اجزاء پر توجہ مرکوز کر کے پیٹرن کی شناخت میں مدد کرتا ہے۔ مسئلہ حل کرنے کے عمل کو ہموار کرتا ہے اور وضاحت کو یقینی بناتا ہے۔
- ڈی کمپوزیشن خاص طور پر سافٹ ویئر ڈویلپمنٹ میں اہم ہے، جہاں بڑے پراجیکٹس کو ماڈیولز میں توڑا جا سکتا ہے۔ جیسے کہ انٹرفیس ڈیزائن، بیک اینڈ ڈویلپمنٹ اور ڈیٹا بیس مینجمنٹ۔ یہ نہ صرف علم کو بہتر بناتا ہے بلکہ اس سے کارکردگی میں بھی اضافہ ہوتا ہے کیوں کہ مختلف اجزاء پر مختلف ٹیمیں یا لوگ بیک وقت کام کر سکتے ہیں۔
- مزید برآں ڈی کنسٹرکشن چھوٹے ذیلی مسائل کے حل کو دوبارہ قابلِ استعمال بناتا ہے کیوں کہ وہ اکثر دوسرے پروجیکٹس یا سٹنگز پر لاگو ہوتے ہیں۔

www.ilmkidunya.com

- یہ پورے نظام کو دوبارہ ڈیزائن کرنے کے بجائے نظاموں کو انفرادی اجزاء میں تبدیلیوں کے ذریعے ارتقا یا موافقت کی اجازت دے کر اسکیل ایبلٹی کو بھی فروغ دیتا ہے۔
- مجموعی طور پر بنیادی مسائل کو حل کرنے، ماڈیولرٹی، کمپیوٹیشنل تھنکنگ اور اس سے آگے کی کارکردگی کے لیے ڈی کمپیوزیشن ایک مؤثر تکنیک ہے

سوال 3: کمپیوٹیشنل تھنکنگ کے حوالے سے پیٹرن کی شناخت کی وضاحت کریں۔ یہ مسئلہ حل کرنے میں کس طرح مدد کرتی ہے؟

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

مثال: مثال کے طور پر، اگر آپ محسوس کرتے ہیں کہ آپ ہمیشہ سوموار کو اپنا ہوم ورک بھول جاتے ہیں تو آپ ایک پیٹرن کو پہچان سکتے ہیں اور خاص طور پر اتوار کے لیے ایک یاد دہانی ترتیب دے سکتے ہیں۔

کمپیوٹیشنل تھنکنگ میں پیٹرن کی شناخت:

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 ہے۔

مثال: آیئے مربعوں کے رقبے میں پیٹرن تلاش کرنے کی مثال پر غور کریں۔ نیچے دی گئی تصویر میں اُوپری قطار مربعوں کی سائیڈ کی لمبائی کی نمائندگی کرتی ہے، جو 1 سے 7 تک ہے۔ نچلی قطار ان مربعوں کے رقبہ کو ظاہر کرتی ہے۔ یہاں ہم ایک نمونہ دیکھ سکتے ہیں کہ کس طرح رقبہ میں اضافہ ہوتا ہے۔

- سائیڈ لمبائی 1: رقبہ $1 = 1^2$
- سائیڈ لمبائی 2: رقبہ $2^2 = 2\ (1 + 3)$
- سائیڈ لمبائی 3: رقبہ $3^2 = 9\ (1 + 3 + 5)$
- سائیڈ لمبائی 4: رقبہ $4^2 = 16 \quad (1 + 3 + 5 + 7)$
- سائیڈ لمبائی 5: رقبہ $5^2 = 25\ (1 + 3 + 5 + 7 + 9)$
- سائیڈ لمبائی 6: رقبہ $6^2 = 36\ (1 + 3 + 5 + 7 + 9 + 11)$
- سائیڈ لمبائی 7: رقبہ $7^2 = 49\ (1 + 3 + 5 + 7 + 9 + 11 + 13)$

وضاحت:

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

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

www.ilmkidunya.com

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۔

جواب: نمونہ: ٹیبل نیچے دیا گیا ہے:

سائیڈ کی لمبائی	رقبہ (اوڈ نمبر)	رقبہ (مربع)
1	1	1
2	4	4
3	9	9
4	16	16
5	25	25
6	36	36
7	49	49
8	64	64
9	81	81
10	100	100

سوال 4: کمپیوٹیشنل تھنکنگ میں Abstraction کیا ہے؟ اس کی اہمیت پر تبادلہ 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: **تجرید (Abstraction):** تجرید پیچیدہ حقیقت کو چھپانے اور صرف ضروری عناصر کو نمایاں کرنے کا عمل ہے۔ یہ ہمیں تفصیلات میں گم ہوئے بغیر اعلیٰ سطح کے جائزہ پر توجہ مرکوز کرتے ہوئے پیچیدگی کو کم کرنے میں مدد کرتا ہے۔

مثال: چائے کا کپ بنانا: بلند سطح کے اقدامات:

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 چینی شامل کریں۔

وضاحت:

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

تجرید (Abstraction) کی اہمیت:

تجرید 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 بنانا، اور غیر ضروری حصوں کو نظر انداز کرنا۔ صرف ضروری تفصیلات پر توجہ مرکوز کرنا شامل ہے۔ اس سے مسائل کو زیادہ موثر طریقے سے سمجھنے، ڈیزائن کرنے اور حل کرنے میں مدد ملتی ہے۔

www.ilmkidunya.com

دلچسپ معلومات:

پیچیدہ مسائل کو حل کرتے وقت انہیں 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۔

سوال 5: 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: الگورتھم: 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 پر عمل کرنے کی طرح۔

HOW TO BAKE A CAKE?

1) Preheat the oven
2) Gather the ingredients
3) Measure out the ingredients
4) Mix together the ingredients to make the batter
5) Grease a pan
6) Pour the batter into the pan
7) Put the pan in the oven
8) Set a timer
9) When the timer goes off, take the pan out of the oven
10) Enjoy!



شکل 7.3: الگورتھم کی مثال: کیک بنانے کی ترکیب

وضاحت:

الگورتھم ایک منظم اور واضح ترتیب میں دی گئی ہدایات کا مجموعہ ہوتا ہے۔ جس پر عمل کر کے کسی خاص مقصد کو حاصل کیا جا سکتا ہے۔ یہ ایک ترکیب یا ہدایات کے ایک سیٹ کی طرح ہے جو آپ کو بتاتا ہے کہ کسی کام کو کس طرح کرنا ہے۔

مثال 1: کیک بنانا: تصویر میں ہم کیک بنانے کی ترکیب دیکھتے ہیں۔ ترکیب اجزا کی ایک فہرست فراہم کرتی ہے اور انھیں ملانے اور کیک پکانے کے لیے مرحلہ وار ہدایات فراہم کرتی ہے۔ یہ الگورتھم کی ایک مثال ہے کیونکہ یہ کیک بنانے کے مقصد کو حاصل کرنے کے لیے اقدامات کی ایک واضح ترتیب کا خاکہ پیش کرتا ہے۔

مثال 2: درخت لگانا: 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، اس بات کو یقینی بنائیں کہ درخت سیدھا ہے۔
4۔ گڑھے کو مٹی سے بھریں، ہوا کو ہٹانے کے لیے اسے آہستہ سے دبائیں۔
5۔ درخت کو دل کھول کر پانی دیں تاکہ وہ مٹی میں جڑ پکڑ سکے۔
6۔ نمی کو برقرار رکھنے کے لیے درخت کی جڑوں کے ارد گرد نامیاتی کھاد ڈالیں۔
7۔ درخت کو باقاعدگی سے پانی دیں جب تک کہ یہ بڑھنا شروع نہ ہو جائے۔

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

سرگرمی 3:

- **آئیں ایک الگورتھم بنائیں!**

کسی ایسے کام کے بارے میں سوچیں جو آپ ہر روز کرتے ہیں، جیسے دانتوں کو برش کرنا یا اسکول بیگ کو پیک تیار کرنا۔ آپ جن اقدامات پر عمل کرتے ہیں انہیں 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!

جواب: عملی کام (پریکٹیکل ورک)

www.ilmkidunya.com

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 4:

- 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۔

الگورتھم چیلنج

- ایک عام کام کے لیے الگورتھم بنانے کے لیے دو دو کے گروپوں میں کام کریں، جیسے سینڈوچ بنانا یا اسکول کے لیے تیار ہونا۔ ہر مرحلہ کو واضح طور پر لکھیں اور پھر دوسرے گروپ کے ساتھ الگورتھم کا تبادلہ کریں۔ ان کے الگورتھم پر بالکل اسی طرح عمل کریں جیسے لکھا گیا ہے اور دیکھیں کہ آیا آپ کام مکمل کر سکتے ہیں۔

جواب: عملی کام (پریکٹیکل ورک)

سوال 6: کمپیوٹیشنل تھنکنگ کے اُصول بیان کریں۔ مثالوں سے وضاحت بھی دیں۔

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

1۔ مسئلہ کی تفہیم

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

> ''اگر میرے پاس کسی مسئلے کو حل کرنے کے لیے ایک گھنٹہ 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۔''
> (البرٹ آئن سٹائن)

مسئلے کی تفہیم کی اہمیت:

- **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۔
- **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۔
- **مؤثر حل:** جب آپ مسئلے کو اچھی طرح سے 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۔
- **غلطیوں سے بچنا:** مسئلے کو اچھی طرح سے سمجھ کر آپ عام نقصانات اور غلطیوں سے بچ سکتے ہیں۔ مسئلے کو غلط سمجھنا غلط اصل اور محنت کے ضائع ہونے کا باعث بن سکتا ہے۔

www.ilmkidunya.com

مثال: اسکول کی ویب سائٹ بنانا

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

- **ضروریات کی شناخت:** ویب سائٹ میں کن خصوصیات کی ضرورت ہے؟ مثال کے طور پر، خبروں اور واقعات، کلاس شیڈول، اور رابطے کی معلومات کے لیے صفحات۔
- **صارف کی ضروریات:** ویب سائٹ کا استعمال کون کرے گا؟ طالب علم، اساتذہ یا والدین؟ اپنے سامعین کو سمجھنے سے ایک صارف دوست انٹر فیس ڈیزائن کرنے میں مدد ملے گی۔
- **تکنیکی رکاوٹیں:** کون سے وسائل اور آلات دستیاب ہیں؟ کیا آپ کو ویب سرور اور ضروری سافٹ ویئر تک رسائی حاصل ہے؟

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

2۔ مسئلے کو آسان بنانا

کسی مسئلے کو آسان بنانے کا مطلب ہے کہ اسے چھوٹے، زیادہ قابل انتظام ذیلی مسائل میں تقسیم کیا جائے۔ مثال کے طور پر: ویب سائٹ ڈیزائن کرنے کے لیے، کاموں کو لے آؤٹ ڈیزائن کرنے، مواد بنانے اور کاموں کو کوڈ کرنے میں تقسیم کریں۔

3۔ 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۔ حل کو ڈیزائن کرنے کے لیے ایک تفصیلی منصوبہ یا الگورتھم بنانے کی ضرورت ہوتی ہے۔

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 7: وضاحت کریں کہ آپ ایک پیچیدہ کمپیوٹیشنل مسئلے کو حل کرنے کے لیے الگورتھم ڈیزائن کے طریقے فلوچارٹ کا استعمال کیسے کریں گے۔ ایک تفصیلی مثال کے ساتھ اپنی وضاحت کو بیان کریں۔

جواب: الگورتھم ڈیزائن کے طریقے: الگورتھم ڈیزائن کے طریقے مختلف کمپیوٹیشنل مسائل سے مؤثر طریقے سے نمٹنے کے لیے مختلف ٹولز اور تکنیک فراہم کرتے ہیں۔ ہر طریقہ کار کی اپنی طاقت اور کمزوریاں ہیں، جو اسے مختلف قسم کے مسائل کے لیے موزوں بناتی ہیں۔ مختلف طریقوں کو سمجھنے سے ایک مسئلے کے لیے سب سے مناسب طریقہ منتخب کرنے میں مدد ملتی ہے، جس سے زیادہ مؤثر اور بہتر حل سامنے آتے ہیں۔

فلوچارٹ:

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

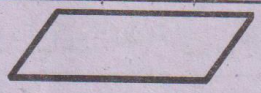
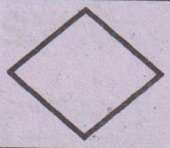
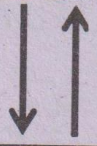
www.ilmkidunya.com

فلوچارٹ کی اہمیت

- **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۔
- **ابلاغ:** یہ پیچیدہ عمل کو وسیع سامعین تک پہنچانے کے لیے بہترین ذریعہ ہیں تاکہ سب کو ایک جیسی سمجھ حاصل ہو۔
- **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 کرنے میں مدد ملتی ہے۔
- **دستاویزات:** یہ نظام اور عمل کے لیے ایک ضروری دستاویزات کے طور پر کام کرتے ہیں جو تربیت اور حوالہ کے مقاصد کے لیے مفید ہوتے ہیں۔

فلوچارٹ کی علامتیں

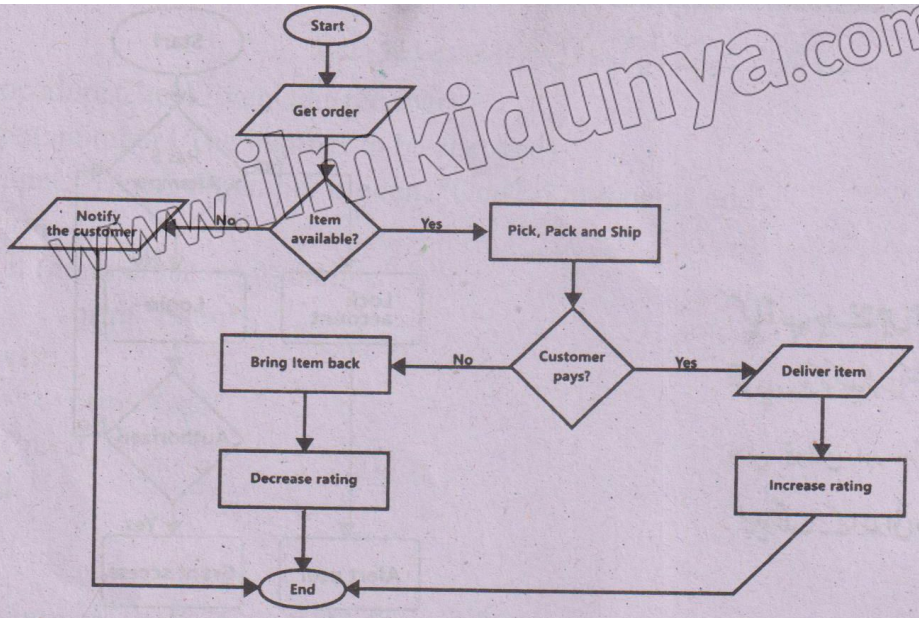
فلوچارٹ کی علامتیں بصری نمائندگی ہوتی ہیں جو کسی عمل یا نظام کے مراحل اور بہاؤ کی وضاحت کرنے کے لیے استعمال ہوتی ہیں جیسا کہ نیچے دیے گئے ٹیبل میں دکھایا گیا ہے۔

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 اسکرین پر آؤٹ پٹ دکھانا۔)	متوازی گرام (Input / Output)	
یہ عمل میں ایک فیصلہ کی جگہ کی نمائندگی کرتا ہے جہاں بہاؤ ہاں یا نہیں، سوال صحیح یا غلط کی بنیاد پر مختلف ہو سکتا ہے۔	ڈائمنڈ (فیصلہ سازی)	
فلوچارٹ 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

مثال 1: آپ کے گھر کے قریب ایک دکان: فرض کریں کہ کوئی دکان موبائل فون پیغامات کے ذریعے آرڈر لیتی ہے۔ شکل 7.4 میں فلوچارٹ آرڈر پروسیسنگ کے مراحل کا خاکہ پیش کرتا ہے۔ ان پٹ آرڈر ہے، اور آؤٹ پٹ آئٹم ڈلیوری ہے اطلاع ہے اگر آئٹم دستیاب نہیں ہے۔ فیصلہ آئٹم کی دستیابی اور گاہک کی ادائیگی کے حوالے سے کیے جاتے ہیں۔ اگر گاہک آئٹم قبول نہیں کرتا ہے یا ادائیگی نہیں کرتا ہے تو، آئٹم کو دکان پر واپس کر دیا جاتا ہے، اور گاہک کی درجہ بندی 1 نمبر کم کر دی جاتی ہے۔ اگر وہ آئٹم کے لیے ادائیگی کرتے ہیں تو گاہک کی درجہ بندی میں 1 نمبر کا اضافہ کر دیا جاتا ہے۔ اگر سامان دستیاب نہیں ہے تو، دکان گاہک کو مطلع کرتی ہے، بصورت دیگر، دکان سامان اٹھاتی ہے، پیک کرتی ہے اور بھیجتی ہے۔

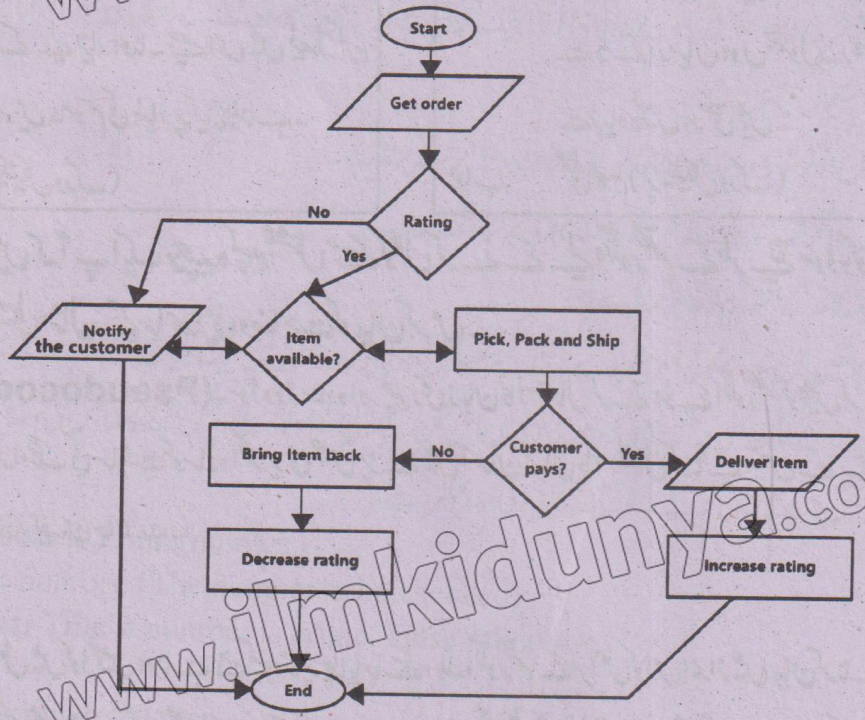
www.ilmkidunya.com



شکل 7.4: دکان کے آرڈر کے عمل کا فلوچارٹ

گاہک کی درجہ بندی کا استعمال کرتے ہوئے فلوچارٹ میں بہتری:

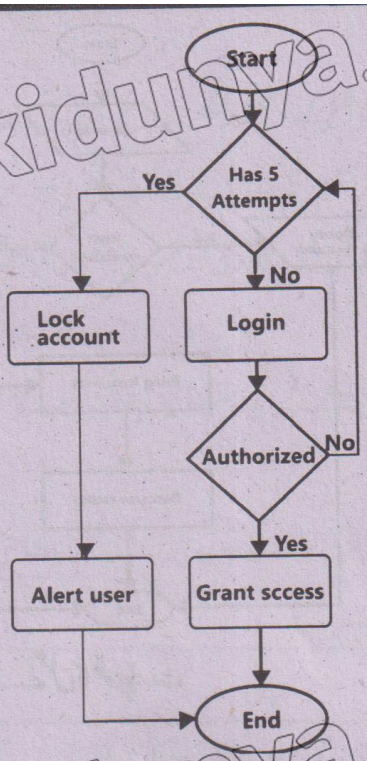
نوٹ کریں کہ اگرچہ گاہک کی درجہ بندی شکل 7.4 میں دکھائے گئے فلوچارٹ میں شامل ہے لیکن اس کا استعمال نہیں کیا گیا ہے۔ آیئے! فلوچارٹ پر نظر ثانی کریں تاکہ یہ یقینی بنایا جاسکے کہ صرف) سے زیادہ درجہ بندی والے صارفین ہی آرڈر کرسکیں۔ نظر ثانی فلوچارٹ نیچے دی گئی شکل میں دکھایا گیا ہے۔



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

مثال 2: لاگ ان سسٹم کے لئے ایک فلوچارٹ، جس میں صارف کا نام اور پاس ورڈ داخل کرنے، اسناد کی تصدیق کرنے، اور رسائی دینے کے مراحل دکھائے گئے ہیں جیسا کہ نیچے دی گئی شکل میں دکھایا گیا ہے۔ یوزر زیادہ سے زیادہ پانچ کوششیں کرسکتا ہے۔

www.ilmkidunya.com



شکل 7.6: لاگ اِن سسٹم کا فلوچارٹ

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۔

سرگرمی 5:

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

جواب: عملی کام (پریکٹیکل ورک)

سرگرمی 6:

- شکل 7.5 کو ترمیم کریں تاکہ صارف کے رینگز درست حد 0 سے 5 کے درمیان ہوں بشمول۔ رینگز منفی نہیں ہو سکتی ہیں یا 5 سے زیادہ نہیں ہو سکتی ہیں۔

جواب: عملی کام (پریکٹیکل ورک)

سوال 8: وضاحت کریں کہ آپ ایک پیچیدہ کمپیوٹیشنل مسئلے کو حل کرنے کے لیے الگورتھم کے طریقے سوڈوکوڈ کا استعمال کیسے کریں گے۔ ایک تفصیلی مثال کے ساتھ اپنی وضاحت کو بیان کریں۔

جواب: سوڈوکوڈ (Pseudocode): سوڈوکوڈ سادہ اور غیر رسمی زبان کا استعمال 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 مفید ذریعہ بن جاتا ہے۔

سوڈوکوڈ کیا ہے؟

سوڈوکوڈ کوئی اصل میں کوڈ نہیں ہوتا ہے جسے کمپیوٹر پر چلایا جا سکے، بلکہ الگورتھم کے مراحل کو اس انداز میں بیان کرنا ہے جسے سمجھنا آسان ہو۔ یہ پروگرامرز اور طالب علموں کو کسی مخصوص پروگرامنگ زبان کے ٹیکسٹ میں پھنسے بغیر الگورتھم کی منطق پر توجہ مرکوز کرنے میں مدد کرتا ہے۔

مثال 1: اس بات کا تعین کرنا کہ آیا کوئی عدد جفت ہے یا طاق، پروگرامنگ اور کمپیوٹر سائنس میں ایک بنیادی کام ہے۔ ایک جفت عدد بغیر کسی باقیات کے 2 سے تقسیم کیا جاتا ہے تو باقی بچ جاتا ہے۔ ذیل میں اس عمل کے لیے سوڈوکوڈ ہے، جس کے بعد ایک وضاحت ہے:

www.ilmkidunya.com

الگورتھم 1: سوڈوکوڈ اس بات کا تعین کرنے کے لیے کہ آیا کوئی نمبر جفت ہے یا طاق ہے۔

```
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:

1۔ **پروسیجر کا اعلان:** سوڈوکوڈ، CheckEvenOdd کی ڈیکلریشن سے شروع ہوتا ہے جو ایک ان پٹ "Number" لیتا ہے۔

2۔ **ان پٹ:** طریقہ کار ایک ویری ایبل "Number" کو قبول کرتا ہے جو چیک کیا جانے والا عدد ہے۔

3۔ **آؤٹ پٹ:** اگر "NumberEven" ہے تو پروسیجر آؤٹ پٹ "Even" دے گا اور اگر Number "Odd" ہے تو پروسیجر آؤٹ پٹ "Odd" دے گا۔

4۔ **آغاز:** پروسیجر کے آغاز کو ظاہر کرتا ہے۔

5۔ **شرط کی جانچ:** شرط if (Number%2 = =0) چیک کرتی ہے کہ کیا 2 پر تقسیم ہونے کے بعد باقی نمبر 0 بچتا ہے۔ اس مقصد کے لیے موڈیول آپریٹر (%) استعمال کیا جاتا ہے۔

6۔ **Even کیس:** اگر شرط درست ہے تو طریقہ کار 'Even' پرنٹ کرتا ہے۔

7۔ **Odd کیس:** اگر شرط غلط ہے تو طریقہ کار 'Odd' پرنٹ کرتا ہے۔

8۔ **اختتام:** طریقہ کار کے اختتام کو ظاہر کرتا ہے۔

مثال 2: اس بات کا تعین کرنا کہ آیا کوئی نمبر مفرد ہے یا نہیں نمبر تھیوری اور کمپیوٹر سائنس میں ایک بنیادی کام ہے۔ ایک مفرد عدد ایک قدرتی عدد ہے جو 1 سے بڑا ہے اور جو اپنے یا کے علاوہ کسی پر تقسیم نہیں ہوتا۔ ذیل میں اس عمل کے لیے سوڈوکوڈ ہے، جس کے بعد جس کی وضاحت ہے۔

الگورتھم 2: سوڈوکوڈ اس بات کا تعین کرنے کے لیے کہ آیا کوئی عدد مفرد ہے۔

```
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
```

www.ilmkidunya.com

```
10:             return False
11:         end if
12:     end for
13:     return True
14: End
```

وضاحت:

1۔ **پروسیجر کا اعلان:** سوڈوکوڈ پروسیجر ''isPrime'' کی ڈیکلریشن سے شروع ہوتا ہے جو ایک ان پٹ ''Number'' لیتا ہے۔

2۔ **ان پٹ:** پروسیجر ایک ویری ایبل ''Number'' کو قبول کرتا ہے جو چیک کیا جانے والا عدد ہے۔

3۔ **آؤٹ پٹ:** اگر نمبر پرائم ہے تو طریقہ کار True اور دوسری صورت میں False' آؤٹ پٹ دے گا۔

4۔ **آغاز:** پروسیجر کے آغاز کو ظاہر کرتا ہے۔

5۔ **ابتدائی چیک:** شرط :(Number < = 1) if چیک کرتی ہے کہ آیا عدد 1 سے کم یا برابر ہے۔ اگر یہ درست ہے تو پروسیجر False' واپس کرتا ہے کیونکہ اسے کم یا اس کے برابر نمبر مفرد عدد نہیں ہیں۔

6۔ **ممکن تقسیم کاروں کی تلاش کے لیے لوپ:** فاز لوپ عدد کی جڑ تک چلتا ہے۔ اس کی وجہ یہ ہے کہ کسی عدد کے بڑے Factors کا اس کے چھوٹے Factors کے ملٹی پل ہوتے ہیں جسے پہلے ہی چیک کیا جا چکا ہے۔

7۔ **تقسیم کی جانچ:** لوپ کے اندر شرط (Number % i = 1) if چیک کرتی ہے کہ آیا عدد i پورے طور پر تقسیم ہو رہا ہے یا نہیں۔ اگر یہ ''True'' ہے تو پروسیجر ''False'' واپس کرتا ہے کیونکہ عدد کا 1 کے علاوہ ایک اور بھی تقسیم کار ہے۔

8۔ **پرائم کنفرمیشن:** اگر لوپ میں کوئی تقسیم کار نہیں پایا جاتا ہے تو، پروسیجر ''True'' واپس کرتا ہے، جس سے اس بات کی تصدیق ہوتی ہے کہ مفرد عدد ہے۔

9۔ **اختتام:** پروسیجر کے اختتام کو ظاہر کرتا ہے۔

سوڈوکوڈ کا استعمال:

سوڈوکوڈ کے استعمال کے متعدد فوائد ہیں:

- **وضاحت:** یہ ٹیکسٹ کے بارے میں فکر کیے بغیر الگورتھم کی منطق کو سمجھنے میں مدد کرتا ہے۔
- **منصوبہ بندی:** یہ پروگرامرز کو اپنے خیالات کا خاکہ پیش کرنے اور الگورتھم کے مراحل کی منصوبہ بندی کرنے میں مدد دیتا ہے۔
- **ابلاغ:** یہ الگورتھم کے مراحل کو بیان کرنے کا ایک عالم گیر طریقہ ہے، جس سے دوسروں کے ساتھ تبادلہ خیال کرنا آسان ہو جاتا ہے۔

سرگرمی 7

- **اپنا سوڈوکوڈ 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 سوڈوکوڈ لکھنے کے لیے کہیں اور پھر اسے کل اس میں پیش کریں۔

جواب: عملی کام (پریکٹیکل ورک)

www.ilmkidunya.com

کیا آپ جانتے ہیں؟

اصل کوڈ لکھنے سے پہلے سافٹ ویئر کی تیاری میں سوڈوکوڈ کو اکثر استعمال کیا جاتا ہے تاکہ 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 پروگرامنگ لینگویجز استعمال کر رہے ہیں۔

سوال 9: الگورتھم ڈیزائن کے طریقوں کے طور پر فلوچارٹ اور سوڈوکوڈ کے طریقوں کا موازنہ کریں۔ ہر طریقہ کار کے فوائد اور نقصانات پر تبادلہ خیال کریں، اور ایسی مثالیں فراہم کریں جہاں ایک کو دوسرے پر ترجیح دی جا سکتی ہے۔

جواب: فلوچارٹ اور سوڈوکوڈ کا موازنہ: فلوچارٹ اور سوڈوکوڈ دونوں الگورتھم کی وضاحت کرنے کے لیے استعمال ہونے والے ذرائع ہیں، لیکن وہ مختلف طریقوں سے ایسا کرتے ہیں۔ ان کے درمیان فرق کو سمجھنے سے آپ کو یہ فیصلہ کرنے میں مدد مل سکتی ہے کہ کس صورت میں کون سا طریقہ استعمال کرنا زیادہ بہتر ہے۔

فلوچارٹ	سوڈوکوڈ
1۔ سوڈوکوڈ الگورتھم کے مراحل کی وضاحت کرنے کے لیے سادہ زبان اور اسٹرکچرڈ فارمیٹ کا استعمال کرتا ہے۔	1۔ فلوچارٹ الگورتھم کے بہاؤ کو ظاہر کرنے کے لیے گرافیکل علامتوں اور تیروں (arrows) کا استعمال کرتے ہیں۔
2۔ یہ ایک کہانی کی طرح پڑھا جاتا ہے، جہاں ہر مرحلہ ترتیب وار لکھا جاتا ہے۔	2۔ یہ ایک فلم دیکھنے کی طرح ہے، جہاں ہر علامت (جیسے مستطیل، ڈائمنڈ اور بیضوی) ایک مختلف قسم کے عمل یا فیصلے کو ظاہر کرتی ہے، اور تیر مرحلوں کے درمیان تعلق اور سمت کی نشاندہی کرتے ہیں۔
3۔ سوڈوکوڈ تفصیلی اور بیانیہ طرز میں الگورتھم کے مراحل کو واضح کرتا ہے۔	3۔ فلوچارٹ ایک بصری انداز میں عمل کو واضح کرتا ہے، جو مجموعی بہاؤ اور ساخت کو سمجھنے کے لیے زیادہ مؤثر ہو سکتا ہے۔
4۔ یہ خاص طور پر الگورتھم کی ایسی دستاویز سازی کے لیے مفید ہے جسے کسی بھی پروگرامنگ لینگویج کے اصل کوڈ میں آسانی سے تبدیل کیا جا سکتا ہے۔	4۔ یہ ایک نظر میں الگورتھم میں شامل مراحل اور فیصلوں کی فوری شناخت کرنے کے لیے مفید ہے۔

ٹیبل 7.2: فلوچارٹ اور سوڈوکوڈ کے درمیان موازنہ

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"
```

www.ilmkidunya.com

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

سوال 10: الگورتھم کی سرگرمیوں کے بارے میں آپ کیا جانتے ہیں؟ وضاحت کریں۔

جواب: الگورتھم کی سرگرمیاں: الگورتھم کی سرگرمیاں درج ذیل ہیں:

ڈیزائن اور تشخیص کی تکنیک

الگورتھم کا جائزہ لینا یہ سمجھنے کے لیے اہم ہے کہ یہ مسائل کو کتنے مؤثر طریقے سے حل کرتے ہیں۔ یہاں ہم الگورتھم کا جائزہ لینے کے لیے مختلف تکنیکوں کو دیکھیں گے، ان کے لیے درکار وقت اور جگہ کی پیچیدگیوں پر توجہ مرکوز کریں گے۔

1۔ ٹائم کمپلیکسٹی

ٹائم کمپلیکسٹی اس بات کی پیمائش کا ایک طریقہ ہے کہ الگورتھم کس قدر تیز یا سست کارکردگی کا مظاہرہ کرتا ہے۔ یہ ہمیں بتاتا ہے کہ ان پٹ کے سائز میں اضافے کے ساتھ الگورتھم کے چلنے کا وقت کیسے تبدیل ہوتا ہے۔ اسے سمجھنے کا ایک آسان طریقہ یہ ہے:

وضاحت:

فرض کریں کہ آپ کے پاس ناموں کی ایک فہرست ہے، اور آپ ایک مخصوص نام تلاش کرنا چاہتے ہیں۔ اگر آپ کے پاس 10 نام ہوں تو آپ کو فہرست کو دیکھنے میں کچھ سیکنڈ لگ سکتے ہیں۔ اگر آپ کے پاس 100 نام ہوں تو یہ وقت کیا ہوگا؟ یا 1,000 نام ہوں تو؟ جیسے جیسے فہرست لمبی ہوتی جائے گی نام تلاش کرنے میں لگنے والا وقت بڑھتا جائے گا۔ درکار وقت کی پیچیدگی ہمیں اس اضافے کو سمجھنے میں مدد دیتی ہے۔

2۔ سپیس کمپلیکسٹی

سپیس کمپلیکسٹی ان پٹ سائز کے لحاظ سے الگورتھم میں استعمال ہونے والی میموری کی مقدار کی پیمائش کرتی ہے۔ ان پٹ کے لیے درکار میموری اور الگورتھم کے ذریعے استعمال ہونے والی کسی بھی اضافی میموری دونوں پر غور کرنا ضروری ہے۔ الگورتھم کو ڈیزائن کرنے اور تشخیص کرنے میں ڈرائی رن اور سیمولیشن جیسی سرگرمیاں شامل ہیں تاکہ یہ یقینی بنایا جا سکے کہ وہ مطلوبہ نتائج فراہم کرتے ہیں۔

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۔

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، 1,000 اشیاء کی فہرست کو ترتیب دینے میں ایک الگورتھم 1 سیکنڈ جب کہ دوسرا الگورتھم 10 سیکنڈ کا وقت لے سکتا 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۔

www.ilmkidunya.com

سرگرمی 8:

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

جواب: عملی کام (پریکٹیکل ورک)

سوال 11: فلوچارٹ اور سوڈوکوڈ دونوں کے حوالے سے ڈرائی رَن کے تصور کی وضاحت کریں۔ ڈرائی رَن کرنے سے الگورتھم کے درست ہونے کی تصدیق کرنے میں کس طرح مدد ملتی ہے؟

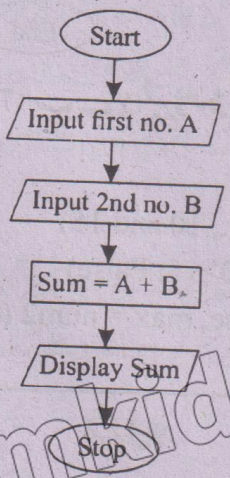
جواب: **ڈرائی رن (Dry Run):** ڈرائی رن میں کسی بھی غلطی کی نشاندہی کرنے کے لیے نمونے کے ڈیٹا کے ساتھ الگورتھم کو دستی طور پر چلایا جاتا ہے۔

فلوچارٹ کا ڈرائی رن:

فلوچارٹ کے ڈرائی رن میں دستی طور پر فلوچارٹ سے قدم بہ قدم چلنا شامل ہے تاکہ یہ سمجھا جا سکے کہ کمپیوٹر کا استعمال کیے بغیر الگورتھم کس طرح کام کرتا ہے۔ اس سے کسی بھی منطقی غلطیوں کی نشاندہی کرنے اور کنٹرول کے بہاؤ کو سمجھنے میں مدد ملتی ہے۔

مثال: دو نمبروں کی جمع دو نمبروں کو جمع کرنے کے لیے نیچے دی گئی تصویر میں دیئے گئے فلوچارٹ پر غور کریں: اس فلوچارٹ کو ڈرائی رن کرنے کے 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



شکل 7.7: دو نمبروں کو جمع کرنے کا فلوچارٹ

سوڈوکوڈ کا ڈرائی رَن:

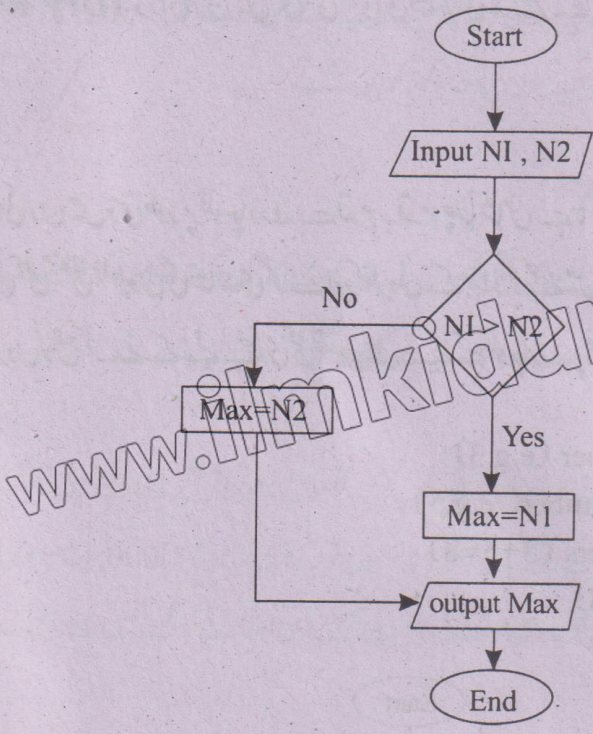
سوڈوکوڈ کے ڈرائی رن کا مطلب اسے لائن بہ لائن دستی طور پر چلا کر اس کے عمل کو سمجھا جائے۔ یہ طریقہ الگورتھم کی منطق اور درستگی کی تصدیق کرنے میں مدد دیتا ہے۔

www.ilmkidunya.com

مثال: سب سے بڑا نمبر ڈھونڈنا: سب سے بڑا نمبر ڈھونڈنے کے لیے نیچے دی گئی تصویر میں دیئے گئے الگورتھم پر غور کریں:

Algorithm 4: 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

سوڈو کوڈ ڈرائی رن کرنے کے مراحل:

1. Input num1 and num2 (e.g., 10 and 15)
2. Check if num1 > num2 (10>15: False)
3. Since the condition is False, max = num2 (max= 15)
4. Output max (15)

سرگرمی 9:

- **ڈرائی رن کا ایک فلوچارٹ**

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۔

جواب: عملی کام (پریکٹیکل ورک)

www.ilmkidunya.com

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 رن کو ڈیبگنگ تکنیک کے طور پر استعمال کرتے ہیں۔

سوال 12: سیمولیشن کے بنیادی مقصد کی وضاحت کریں۔ اس کے فوائد لکھیں اور مثالیں دیں۔

جواب: سیمولیشن (Simulation): سیمولیشن ایک ایسا طریقہ ہے جس میں ہم کمپیوٹر پروگراموں کو حقیقی دنیا کے عمل یا نظام کا ماڈل بنانے کے لیے استعمال کرتے ہیں۔ اس سے ہمیں یہ سمجھنے میں مدد ملتی ہے کہ حقیقی زندگی میں انھیں آزمانے کی ضرورت کے بغیر مختلف الگورتھمز کی جانچ کر کے چیزیں کس طرح کام کرتی ہیں۔

سیمولیشن کا استعمال:

1. **ٹیسٹنگ الگورتھم:** ہم یہ دیکھنے کے لیے سیمولیشن کا استعمال کر سکتے ہیں کہ ایک الگوتھم مختلف قسم کے ڈیٹا کے ساتھ کتنی اچھی طرح کام کرتا ہے۔ مثال کے طور پر، اگر ہم نمبروں کو ترتیب دینے کا ایک نیا طریقہ آزمانا چاہتے ہیں تو، ہم اسے نمبروں کے مختلف سیٹوں کے ساتھ سیمولیٹ کر سکتے ہیں تاکہ یہ دیکھا جا سکے کہ یہ کتنا تیز اور صحیح ہے۔
2. **ممکنہ حالات کا تجزیہ:** سیمولیشن ہمیں مختلف حالات پیدا کرنے کی اجازت دیتا ہے تاکہ معلوم ہو سکے کہ ان تبدیلیوں سے کیا نتائج برآمد ہوتے ہیں۔ مثال کے طور پر پودوں کی نشوونما کے بارے میں ایک سائنسی تجربے میں ہم پانی یا سورج کی روشنی کی مختلف مقدار کو سیمولیٹ کر سکتے ہیں تاکہ یہ معلوم کیا جا سکے کہ کون سے حالات پودوں کی نشوونما میں بہترین مدد کرتے ہیں۔

سیمولیشن کے فوائد:

- **کم لاگت:** حقیقی تجربات کرنے کے مقابلے میں سیمولیشن چلانا عموماً سستا اور تیز تر ہوتا ہے۔
- **محفوظ:** ہم کسی کو بھی خطرے میں ڈالے بغیر عمارت میں آگ جیسے خطرناک حالات کی جانچ کر سکتے ہیں۔
- **دہرانے کے قابل:** ہم ایک ہی سیمولیشن کو مختلف ترتیب کے ساتھ کئی بار چلا سکتے ہیں یہ دیکھنے کے لیے کہ چیزیں کس طرح تبدیل ہوتی ہیں۔

سیمولیشن کی مثالیں:

1. **موسم کی پیش گوئی:** ماہرین موسمیات موسم کی پیش گوئی کرنے کے لیے سیمولیشن کا استعمال کرتے ہیں۔ وہ درجہ حرارت نمی اور ہوا کی رفتار کے بارے میں ڈیٹا کو ایک کمپیوٹر ماڈل میں ان پٹ کرتے ہیں تاکہ یہ دیکھا جا سکے کہ اگلے چند دنوں میں موسم کس طرح تبدیل ہو سکتا ہے۔
2. **ٹریفک کا بہاؤ:** شہر کے منصوبہ ساز ٹریفک کی نقل کر سکتے ہیں یہ دیکھنے کے لیے کہ سڑکوں یا ٹریفک لائٹس میں تبدیلیاں گاڑیوں کے بہاؤ کو کس طرح متاثر کر سکتی ہیں۔ اس سے انہیں سڑکوں کو بہتر ڈیزائن کرنے اور ٹریفک جام کو کم کرنے میں مدد ملتی ہے۔

سوال 13: LARP کیا ہے؟ الگورتھم سیکھنے اور مشق کرنے میں اس کی اہمیت پر تبادلہ خیال کریں۔

جواب: لاجک آف الگورتھمز فار ریزولیوشن آف پرابلمز (LARP): LARP کا مطلب مسئلے کے حل کے لیے الگورتھم کی منطق ہے۔ یہ سیکھنے کا ایک دلچسپ اور انٹرایکٹو طریقہ ہے کہ الگورتھم کو عملی طور پر چلا کر ان کے نتائج دیکھے جا سکتے ہیں۔ اسے کھیل کے میدان کے طور پر سوچیں جہاں آپ مختلف الگورتھم کے ساتھ تجربہ کر سکتے ہیں اور سمجھ سکتے ہیں کہ وہ ڈیٹا پر کس طرح عمل کرتے ہیں۔

www.ilmkidunya.com

LARP کی اہمیت:

- یہ جانیں کہ الگورتھم کیسے کام کرتا ہے۔ مثال کے طور پر نیچے دی گئی تصویر کو دیکھیں جو ایک الگورتھم کی وضاحت کرتی ہے جو کسی شخص کی سالانہ تنخواہ پر ٹیکس کے اطلاق کا تعین کرنے کے لیے بنایا گیا ہے۔
- مختلف ان پٹ کے آؤٹ پٹ پر اثرات کا جائزہ لیں۔
- اپنے الگورتھم لکھنے اور انھیں بہتر بنانے کی مشق کریں۔

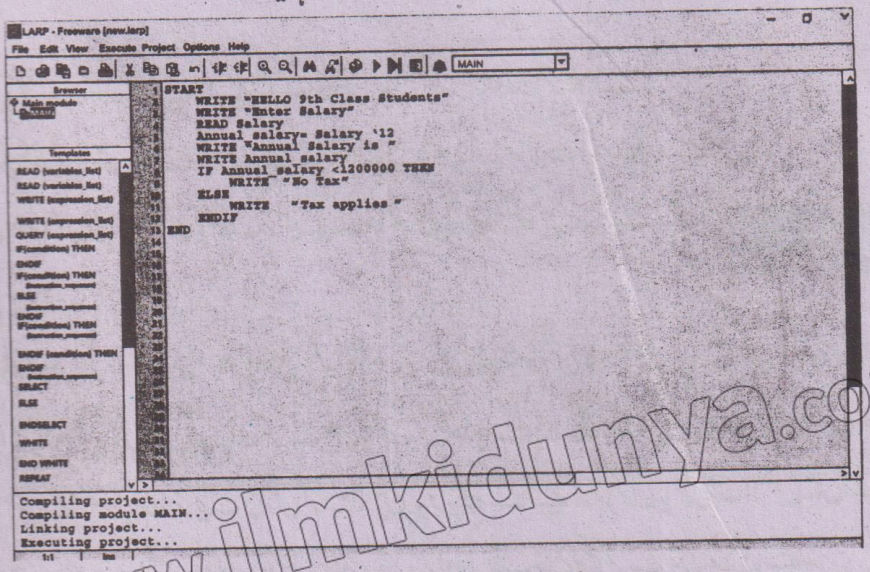
الگورتھم لکھنا:

LARP کا استعمال کرتے ہوئے الگورتھم لکھنا آسان ہے جس کے ذریعے مسائل کے لیے منطقی حل تیار کیے جاسکتے ہیں۔ LARP ایک واضح ٹیکسٹ استعمال کرتا ہے جو اسٹارٹ کمانڈ سے شروع ہوتا ہے اور اینڈ کمانڈ کے ساتھ ختم ہوتا ہے، اس بات کو یقینی بناتا ہے کہ الگورتھم کے ہر مرحلے پر عمل کرنا آسان ہو۔ اس فریم ورک کے اندر، ہدایات سیدھے سادے انداز میں فراہم کی جاتی ہیں، جیسے پیغامات کو ظاہر کرنے کے لیے write کا استعمال کرنا، ان پٹ قیمتوں کے لیے read اور فیصلہ سازی کے لیے if-then-else کے جیسی سٹیٹمنٹس استعمال کرتے ہیں۔

الگورتھم پڑھنا:

پیچیدہ مسائل کو قابل انتظام مراحل میں تقسیم کرنے سے LARP سیکھنے والوں کو پیچیدہ کوڈنگ کے ٹیکسٹ پر پھنسے بغیر الگورتھم کے منطقی بہاؤ پر توجہ مرکوز کرنے کی اجازت دیتا ہے۔ یہ طریقہ نہ صرف الگورتھم ڈیزائن کے بنیادی تصورات کو سمجھنے میں مدد دیتا ہے بلکہ واضح اور منطقی تھنکنگ کی حوصلہ افزائی کر کے مسئلہ حل کرنے کی مہارت کو بھی بڑھاتا ہے۔

مثال: یہاں ایک سادہ الگورتھم کی مثال ہے یہ چیک کرنے کے لیے کہ آیا کوئی نمبر جفت ہے یا طاق:



شکل 7.9: LARP سافٹ ویئر

```
START
        WRITE "Enter a number"
        READ number
        IF number % 2 == 0 THEN
                WRITE "The number is even"
```

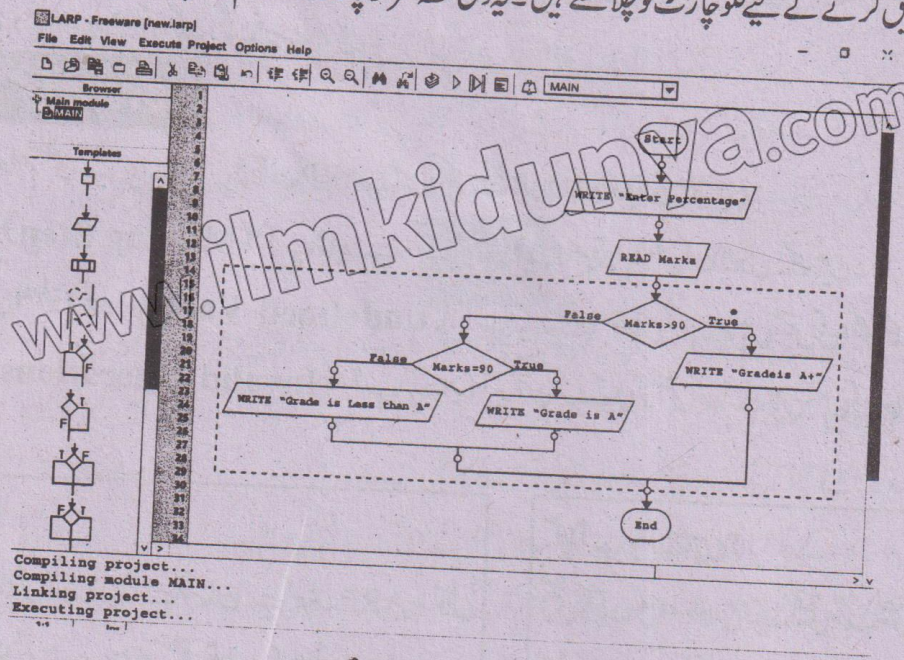
www.ilmkidunya.com

```
        WRITE "The number is odd"
    ENDIF
END
```

LARP میں فلوچارٹ بنانا:

LARP میں فلوچارٹ بنانا الگورتھم کے مراحل کو بھری طور پر پیش کرنا شامل ہے جس میں فلوچارٹ کی علامتوں جیسا کہ پروسیسنگ کے لیے مستطیل، فیصلوں کے لیے ڈائمنڈ اور ان پٹ آؤٹ پٹ آپریشنز کے لیے پیرالیلوگرامز شامل ہیں۔ ایک بار فلوچارٹ بنانے کے بعد، اس فلوچارٹ کو LARP کے ٹیکسٹ میں ترجمہ کر کے LARP میں چلایا جاتا ہے، جس میں else if, if, Read, Write, Start جیسے سیدھے احکامات کا استعمال ہوتا ہے۔ یہ عمل طالب علموں کو اپنے الگورتھم کی منطق کا تصور کرنے اور اس کے قدم بہ قدم عملدرآمد کو دیکھنے کی اجازت دیتا ہے۔

مثال: مثال کے طور پر نیچے دی گئی شکل کسی طالب علم کے گریڈ کو "A" سے اُوپر کرنے یا نہ کرنے کا تعین کرنے کے لیے ایک فلوچارٹ دکھاتا ہے۔ ہم اس کے درست ہونے کی تصدیق کرنے کے لیے فلوچارٹ کو چلا سکتے ہیں۔ یہ دستی نقطہ نظر فلوچارٹ کے کام کے بارے سیکھنے اور جاننے میں مدد دیتا ہے۔



شکل LARP:7.10 میں فلوچارٹ

سوال LARP:14 لکھتے وقت غلطیوں کی شناخت اور ڈیبگنگ کے بارے میں آپ کیا جانتے ہیں؟ وضاحت کریں۔

جواب: غلطیوں کی شناخت اور ڈیبگنگ: جب ہم LARP میں الگورتھم لکھتے ہیں یا فلوچارٹ بناتے ہیں تو ہم کبھی کبھی غلطیاں کرتے ہیں جنہیں غلطیاں یا بگز کہا جاتا ہے۔ یہ غلطیاں ہمارے الگورتھم کو صحیح طریقے سے کام کرنے سے روک سکتی ہیں۔ غلطی سے نمٹنے اور ڈیبگنگ وہ عمل ہیں جو ہمیں ان غلطیوں کو تلاش کرنے اور ٹھیک کرنے میں مدد کرتے ہیں۔

غلطیوں کی اقسام شکل:

آپ کو تین اہم قسم کی غلطیوں کا سامنا کرنا پڑ سکتا ہے:

1۔ **سینٹکس (Syntex) کی غلطیاں:** یہ اس وقت ہوتی ہیں جب ہم اپنے الگورتھم یا فلوچارٹ میں کچھ غلط لکھتے ہیں۔ مثال کے طور پر ایک سٹیپ کو چھوڑ دینا یا غلط علامت کا استعمال کرنا۔

www.ilmkidunya.com

2۔ **رن ٹائم (Run-time) غلطیاں:** یہ اس وقت ہوتی ہیں جب الگورتھم یا فلوچارٹ پر عمل درآمد کیا جا رہا ہوتا ہے۔ مثال کے طور پر، صفر سے تقسیم کرنے جیسے ناممکن آپریشن کو انجام دینے کی کوشش کرنا۔

3۔ **منطقی (Logical) غلطیاں:** یہ الگورتھم کی منطقی غلطیاں ہیں۔ مثال کے طور پر فیصلے کرنے والے حلے پر غلط سمبلز کا استعمال کرنا۔

ڈیبگنگ کی تکنیکیں:

ڈیبگنگ الگورتھم یا فلوچارٹ میں غلطیوں کو تلاش کرنے اور ٹھیک کرنے کا عمل ہے۔ یہاں کچھ عام ڈیبگنگ کی تکنیک کے بارے میں جانتے ہیں:

- **مراحل کا جائزہ لیں:** اپنے الگورتھم یا فلوچارٹ کے ہر مرحلے سے گزریں اور دیکھیں کہ یہ کہاں غلط ہوتا ہے۔
- **کمنٹس استعمال کریں:** اپنے الگورتھم میں کمنٹس یا نوٹ لکھیں تا کہ یہ وضاحت کی جا سکے کہ ہر حصے کو کیا کرنا چاہیے۔ اس سے آپ کو غلطیوں کی نشاندہی کرنے میں مدد مل سکتی ہے۔
- **شرائط کی جانچ کریں:** اس بات کو یقینی بنائیں کہ فیصلہ سازی کے تمام مراحل میں شرائط درست ہیں۔ مسئلے کو آسان بنائیں: الگوتھیم کو چھوٹے حصوں میں توڑ دیں اور ہر حصے کو الگ الگ ٹیسٹ کریں۔

LARP میں عام غلطی کے پیغامات:

یہاں کچھ عام غلطی کے پیغامات ہیں جو آپ LARP میں دیکھ سکتے ہیں اور ان کو جان سکتے ہیں:

- **گمشدہ مرحلہ (Missing Step):** آپ شاید اپنے الگورتھم میں ایک اہم قدم شامل کرنا بھول گئے ہیں۔
- **غیر متعینہ وری ایبلو (Undefined Variables):** آپ ایک متغیر استعمال کر رہے ہیں جس کی ابھی تک وضاحت نہیں کی گئی ہے۔
- **غلط آپریشن (Invalid Operations):** آپ ایک ایسا آپریشن کرنے کی کوشش کر رہے ہیں جس کی اجازت نہیں ہے، جیسے صفر سے تقسیم کرنا۔

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۔

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۔

دلچسپ معلومات:

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

سرگرمی 9:

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

جواب: عملی کام (پریکٹیکل ورک)

www.ilmkidunya.com

سوال 15: LARP کمپیوٹیشنل تھنکنگ کے اُصولوں 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: الگورتھم کو بہتر بنانے کے لیے LARP کا استعمال: LARP (مسائل کے حل کے لیے الگورتھم کی منطق) مسائل کے حل کے لیے ایک عمیق، متعامل اور باہمی تعاون پر مبنی نقطہ نظر اپنا کر کمپیوٹیشنل سوچ کے اُصولوں کے علم اور نفاذ کو بہتر بناتا ہے۔

وضاحت:

صارفین کو پیچیدہ مسائل تحلیل کرنے، متعلقہ معلومات کا خلاصہ کرنے اور کردار ادا کرنے والے حالات کے ذریعے کامیاب الگورتھم ڈیزائن کرنے کا تجربہ حاصل ہوتا ہے، جو حقیقی دُنیا کے چیلنجوں کو نقل کرتے ہیں۔ LARP فوری فیڈ بیک حاصل کرتے ہوئے شرکا کو فعال طور پر حل تخلیق کرنے اور جانچنے کی ضرورت کے ذریعے تجریدی موضوعات کو ٹھوس بناتا ہے۔ جس سے بڑھتی ہوئی بہتری کی حوصلہ افزائی ہوتی ہے۔ یہ حکمتِ عملی تعاون اور موافقت پر مرکوز ہے، جو کمپیوٹیشنل مسئلہ حل کرنے کی متحرک نوعیت کی عکاسی کرتی ہے۔

مثال: مثال کے طور پر ایک کیس پر غور کریں جس میں LARP کا استعمال اونچائی کی بنیاد پر لائبریری میں جلدوں کو ترتیب دینے کے لیے ترتیب والا الگورتھم تیار کرنے کے لیے کیا جاتا ہے۔ صارفین ''لائبریرین''، ''کتابیں'' اور ''مبصرین'' جیسے کردار ادا کرتے ہیں۔ لائبریرین، صارفین (کتابوں) کو ان کے ظاہر کردہ اُونچائی کے مطابق ترتیب دینے کے لیے ایک ابتدائی الگورتھم جیسے ببل کی ترتیب کو استعمال کرتا ہے۔

خلاصہ

- 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۔
- ڈی کمپوزیشن ایک پیچیدہ مسئلے کو چھوٹے، زیادہ منظم حصوں میں توڑنے کا عمل ہے۔
- پیٹرن کی شناخت میں مسائل کے درمیان اور ان کے اندر مماثلت یا نمونے تلاش کرنا شامل ہے۔
- 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۔
- الگورتم کسی مسئلے کو حل کرنے یا کسی کام کو مکمل کرنے کے لیے ہدایات کا مرحلہ وار مجموعہ ہے۔
- مسئلے کو سمجھنا مسئلے کو حل کرنے میں پہلا اور سب سے اہم مرحلہ ہے، خاص طور پر کمپیوٹیشنل تھنکنگ میں۔
- کسی مسئلے کو آسان بنانے میں اسے چھوٹے، زیادہ منظم ذیلی مسائل میں توڑنا شامل ہے۔
- بہترین حل کا انتخاب کرنے میں مختلف طریقوں کا جائزہ لینا اور سب سے زیادہ موثر کا انتخاب کرنا شامل ہے۔
- فلوچارٹ کسی عمل یا نظام میں مراحل کی بصری نمائندگی ہیں، جو تیروں سے منسلک مختلف علامتوں کا استعمال کرتے ہوئے دکھایا گیا ہے۔
- 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۔
- ٹائم کمپلیکسٹی اس بات کی پیمائش کرنے کا ایک طریقہ ہے کہ الگورتھم کتنی تیز یا سست کارکردگی کا مظاہرہ کرتا ہے۔ یہ ہمیں بتاتا ہے کہ ان ریاست پٹ کے سائز میں اضافے کے ساتھ الگورتھم کا چلنے کا وقت کیسے تبدیل ہوتا ہے۔

www.ilmkidunya.com

- سپیس کمپلیکسٹی ان پٹ سائز کے ساتھ الگورتھم استعمال ہونے والی میموری کی مقدار کی پیمائش 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۔
- 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۔
- سیمولیشن اس وقت ہوتا ہے جب ہم حقیقی دنیا کے عمل یا نظام کا ماڈل بنانے کے لیے کمپیوٹر پروگراموں کا استعمال کرتے ہیں۔
- 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۔
- ڈی بگنگ 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) تجرید (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

www.ilmkidunya.com

(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 کوڈ 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) سوڈو کوڈ 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 ڈیبگنگ 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 سوڈو کوڈ 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)

2۔ مختصر سوالات۔

(i) کمپیوٹیشنل تھنکنگ کی وضاحت کریں۔

جواب: کمپیوٹیشنل تھنکنگ (CT) کسی مسئلہ کو حل کرنے کا ایک عمل ہے جس میں پیچیدہ مسائل کو اس طرح حل کرنے کے لیے مہارتوں اور تکنیکوں کا ایک مجموعہ شامل ہے جس پر کمپیوٹر بھی عمل درآمد کر سکتا ہے۔ اس طریقہ کار کو کمپیوٹر سائنس سے باہر بھی مختلف شعبوں میں استعمال کیا جا سکتا ہے، جیسے بیالوجی، ریاضی اور یہاں تک کہ روزمرہ کی زندگی۔

(ii) کمپیوٹیشنل تھنکنگ میں Decomposition کیا ہے؟

جواب: ڈی کمپیوزیشن (Decomposition) یا تقسیم ایک پیچیدہ مسئلے کو چھوٹے، زیادہ منظم حصوں میں تقسیم کرنے کا عمل ہے۔ مثال کے طور پر، اسکول کی تقریب کی منصوبہ بندی کرتے وقت، آپ کام کو کچھ حصوں میں تقسیم کر سکتے ہیں جیسے تاریخ کا انتخاب کرنا، مقام کی بکنگ کرنا، دعوت نامے بھیجنا اور سرگرمیوں کا اہتمام کرنا۔

www.ilmkidunya.com

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

جواب: 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 آپ ایک پیٹرن کو پہچان سکتے ہیں اور خاص طور پر اتوار کے لیے ایک یاد دہانی ترتیب دے سکتے ہیں۔

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

جواب: تجدید پیچیدہ حقیقت کو چھپانے اور صرف ضروری عناصر کو نمایاں کرنے کا عمل ہے۔ یہ ہمیں تفصیلات میں گم ہوئے بغیر اعلیٰ سطح کے جائزہ پر توجہ مرکوز کرتے ہوئے پیچیدگی کو کم کرنے میں مدد کرتا ہے۔ 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 بنانا، اور غیر ضروری حصوں کو نظر انداز کرنا۔ صرف ضروری تفصیلات پر توجہ مرکوز کرنا شامل ہے۔ اس سے مسائل کو زیادہ موثر طریقے سے سمجھنے، ڈیزائن کرنے اور حل کرنے میں مدد ملتی ہے۔

(v) الگورتھم کیا ہے؟

جواب: 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 پر عمل کرنے کی طرح۔ الگورتھم ایک منظم اور واضح ترتیب میں دی گئی ہدایات کا مجموعہ ہوتا ہے۔ جس پر عمل کر کے کسی خاص مقصد کو حاصل کیا جا سکتا ہے۔ یہ ایک ترکیب یا ہدایات کے ایک سیٹ کی طرح ہے جو آپ کو بتاتا ہے کہ کسی کام کو کس طرح کرنا ہے۔

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

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

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

جواب: فلوچارٹ کسی عمل یا نظام کے مراحل کی بصری نمائندگی ہوتے ہیں، جو تیروں سے منسلک مختلف علامتوں کا استعمال کرتے ہوئے بنائے جاتے ہیں۔ یہ وسیع پیمانے پر مختلف شعبوں میں استعمال ہوتے ہیں، بشمول کمپیوٹر سائنس، انجینئرنگ، کاروبار تاکہ عمل کو ماڈل کیا جا سکے، نظاموں کو ڈیزائن کیا جا سکے اور پیچیدہ ورک فلو کو واضح اور موثر انداز میں بیان کی جا سکے۔

(viii) سوڈو کوڈ کے مقصد کی وضاحت کریں۔

جواب: سوڈو کوڈ سادہ اور غیر رسمی زبان کا استعمال 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 مفید ذریعہ بن جاتا ہے۔ سوڈو کوڈ کوئی اصل میں کوڈ نہیں ہوتا ہے جسے کمپیوٹر پر چلایا جا سکے، بلکہ الگورتھم کے مراحل کو اس انداز میں بیان کرنا ہے جسے سمجھنا آسان ہو۔ یہ پروگرامرز اور طالب علموں کو کسی مخصوص پروگرامنگ زبان کے ٹیکسٹ میں پھنسے بغیر الگورتھم کی منطق پر توجہ مرکوز کرنے میں مدد کرتا ہے۔

www.ilmkidunya.com

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

جواب: فلوچارٹ اور سوڈوکوڈ میں فرق درج ذیل ہے:

فلوچارٹ	سوڈوکوڈ
1۔ سوڈوکوڈ الگورتھم کے مراحل کی وضاحت کرنے کے لیے سادہ زبان اور اسٹرکچرڈ فارمیٹ کا استعمال کرتا ہے۔	1۔ فلوچارٹ الگورتھم کے بہاؤ کو ظاہر کرنے کے لیے گرافیکل علامتوں اور تیروں (arrows) کا استعمال کرتے ہیں۔
2۔ یہ ایک کہانی کی طرح پڑھا جاتا ہے، جہاں ہر مرحلہ ترتیب وار لکھا جاتا ہے۔	2۔ یہ ایک فلم دیکھنے کی طرح ہے، جہاں ہر علامت (جیسے مستطیل، ڈائمنڈ اور بیضوی) ایک مختلف قسم کے عمل یا فیصلے کو ظاہر کرتی ہے، اور تیر مرحلوں کے درمیان تعلق اور سمت کی نشاندہی کرتے ہیں۔
3۔ سوڈوکوڈ تفصیلی اور بیانیہ طرز میں الگورتھم کے مراحل کو واضح کرتا ہے۔	3۔ فلوچارٹ ایک بصری انداز میں عمل کو واضح کرتا ہے، جو مجموعی بہاؤ اور ساخت کو سمجھنے کے لیے زیادہ مؤثر ہو سکتا ہے۔
4۔ یہ خاص طور پر الگورتھم کی ایسی دستاویز سازی کے لیے مفید ہے جسے کسی بھی پروگرامنگ لینگویج کے اصل کوڈ میں آسانی سے تبدیل کیا جا سکتا ہے۔	4۔ یہ ایک نظر میں الگورتھم میں شامل مراحل اور فیصلوں کی فوری شناخت کرنے کے لیے مفید ہے۔

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

جواب: ڈرائی رن میں کسی بھی غلطی کی نشاندہی کرنے کے لیے نمونے کے ڈیٹا کے ساتھ الگورتھم کو دستی طور پر چلایا جاتا ہے۔

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

جواب: LARP کا مطلب مسئلے کے حل کے لیے الگورتھم کی منطق ہے۔ یہ سیکھنے کا ایک دلچسپ اور انٹرایکٹو طریقہ ہے کہ الگورتھم کو عملی طور پر چلا کر ان کے نتائج دیکھے جا سکتے ہیں۔ اسے کھیل کے میدان کے طور پر سوچیں جہاں آپ مختلف الگورتھم کے ساتھ تجربہ کر سکتے ہیں اور سمجھ سکتے ہیں کہ وہ ڈیٹا پر کس طرح عمل کرتے ہیں۔

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

جواب: ڈیبگنگ الگورتھم یا فلوچارٹ میں غلطیوں کو تلاش کرنے اور ٹھیک کرنے کا عمل ہے۔ یہاں کچھ عام ڈیبگنگ کی تکنیک کے بارے میں جانتے ہیں:

- **مراحل کا جائزہ لیں:** اپنے الگورتھم یا فلوچارٹ کے ہر مرحلے سے گزریں اور دیکھیں کہ یہ کہاں غلط ہوتا ہے۔
- **کمنٹس استعمال کریں:** اپنے الگورتھم میں کمنٹس یا نوٹ لکھیں تاکہ یہ وضاحت کی جا سکے کہ ہر حصے کو کیا کرنا چاہیے۔ اس سے آپ کو غلطیوں کی نشاندہی کرنے میں مدد مل سکتی ہے۔
- **شرائط کی جانچ کریں:** اس بات کو یقینی بنائیں کہ فیصلہ سازی کے تمام مراحل میں شرائط درست ہیں۔. مسئلے کو آسان بنائیں: الگوتھیم کو چھوٹے حصوں میں توڑ دیں اور ہر حصے کو الگ الگ ٹیسٹ کریں۔

www.ilmkidunya.com

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:

Start

Input: Marks obtained by the student (let's call it marks).
If marks >= 90, assign grade as A+.
Else if 80 <= marks < 90, assign grade as A.
Else if 70 <= marks < 80, assign grade as B.
Else if 60 <= marks < 70, assign grade as C.
Else assign grade as F.

Output: Display the assigned grade.

End

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

جواب: سوال نمبر 7 کا جواب ملاحظہ کریں۔

(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: سوال نمبر 1 کا جواب ملاحظہ کریں۔

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

جواب: سوال نمبر 2 کا جواب ملاحظہ کریں۔

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

جواب: سوال نمبر 3 کا جواب ملاحظہ کریں۔

(vi) کمپیوٹیشنل تھنکنگ میں Abstraction کیا ہے؟ اس کی اہمیت پر تبادلہ 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: سوال نمبر 4 کا جواب ملاحظہ کریں۔

www.ilmkidunya.com

(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: سوال نمبر 5 کا جواب ملاحظہ کریں۔

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

جواب: سوال نمبر 9 کا جواب ملاحظہ کریں۔

(ix) فلوچارٹ اور سوڈوکوڈ 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: سوال نمبر 11 کا جواب ملاحظہ کریں۔

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

جواب: سوال نمبر 13 کا جواب ملاحظہ کریں۔

(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: سوال نمبر 15 کا جواب ملاحظہ کریں۔

اضافی کثیر الانتخابی سوالات (بمطابق حاصلاتِ تعلّم)

✪ درست جواب پر (✓) کا نشان لگائیں۔

-1 یہ 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:

(الف) الگورتھم (ب) ڈی کمپوزیشن (ج) کمپیوٹیشنل تھنکنگ (د) ڈرائی رن

-2 یہ ایک پیچیدہ مسئلے کو چھوٹے، زیادہ منظم حصوں میں توڑنے کا عمل ہے:

(الف) الگورتھم (ب) ڈی کمپوزیشن (ج) کمپیوٹیشنل تھنکنگ (د) سیمولیشن

-3 اس میں مسائل کے درمیان اور ان کے اندر مماثلت یا نمونے تلاش کرنا شامل ہے:

(الف) پیٹرن کی شناخت (ب) تجدید (ج) ڈی کمپوزیشن (د) ڈرائی رن

-4 اس میں پیچیدہ مسائل کو چھوٹے اور زیادہ منظم حصوں میں تقسیم کرنے کو آسان بنانا شامل ہے:

(الف) پیٹرن کی شناخت (ب) تجدید (ج) ڈی کمپوزیشن (د) سیمولیشن

-5 یہ کسی مسئلے کو حل کرنے یا کسی کام کو مکمل کرنے کے لیے ہدایات کا مرحلہ وار مجموعہ ہے:

(الف) الگورتھم (ب) ڈی کمپوزیشن (ج) کمپیوٹیشنل تھنکنگ (د) ڈرائی رن

-6 مسئلے کو حل کرنے میں پہلا اور سب سے اہم مرحلہ ہے، خاص طور پر کمپیوٹیشنل تھنکنگ میں:

(الف) تجدید (ب) مسئلے کا انتخاب (ج) مسئلے کو آسان بنانا (د) مسئلے کو سمجھنا

www.ilmkidunya.com

-7 کسی مسئلے کو _____ میں اسے چھوٹے، زیادہ منظم ذیلی مسائل میں توڑنا شامل ہے:

(الف) حل کرنے میں (ب) آسان بنانے (ج) سمجھنے میں (د) ختم کرنے میں

-8 یہ کسی عمل یا نظام میں مراحل کی بصری نمائندگی ہیں، جو تیروں سے منسلک مختلف علامتوں کا استعمال کرتے ہوئے دکھایا گیا ہے:

(الف) سوڈوکوڈ (ب) سسٹم (ج) فلوچارٹ (د) ڈیٹا بیس

-9 یہ سادہ اور غیر رسمی زبان کا استعمال کرتے ہوئے الگورتھم کی نمائندگی کرنے کا ایک طریقہ ہے جسے سمجھنا آسان ہے:

(الف) سوڈوکوڈ (ب) سسٹم (ج) فلوچارٹ (د) ڈیٹا بیس

-10 یہ پروگرامنگ زبانوں کی ساخت کو سادہ انگریزی زبان پڑھنے کی قابلیت کے ساتھ جوڑتا ہے:

(الف) سوڈوکوڈ (ب) سسٹم (ج) فلوچارٹ (د) ڈیٹا بیس

-11 یہ اس بات کی پیائش کرنے کا ایک طریقہ ہے کہ الگورتھم کتنی تیز یا ست کارکردگی کا مظاہرہ کرتا ہے:

(الف) ٹائم کمپلیکسٹی (ب) سپیس کمپلیکسٹی (ج) الف اور ب دونوں (د) کوئی بھی نہیں

-12 یہ ان پٹ سائز کے سلسلے میں الگورتھم استعمال ہونے والی میموری کی مقدار کی پیائش کرتی ہے:

(الف) ٹائم کمپلیکسٹی (ب) سپیس کمپلیکسٹی (ج) الف اور ب دونوں (د) کوئی بھی نہیں

-13 اس 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:

(الف) سیمولیشن (ب) ڈی کمپوزیشن (ج) کمپیوٹیشنل تھنکنگ (د) ڈرائی رن

-14 یہ اس وقت ہوتا ہے جب ہم حقیقی دنیا کے عمل یا نظام کا ماڈل بنانے کے لیے کمپیوٹر پروگراموں کا استعمال کرتے ہیں:

(الف) سیمولیشن (ب) ڈی کمپوزیشن (ج) کمپیوٹیشنل تھنکنگ (د) ڈرائی رن

-15 اس کا مطلب ہے لائیو الگورتھم چلانا اور مشق کرنا:

(الف) ڈرائی رن (ب) LARP (ج) LAPR (د) سیمولیشن

-16 یہ الگورتھم یا فلوچارٹ میں غلطیوں کو تلاش کرنے اور ٹھیک کرنے کا عمل ہے:

(الف) ڈرائی رن (ب) سیمولیشن (ج) ڈی کمپوزیشن (د) ڈی بگنگ

-17 یہ اس وقت ہوتی ہیں جب ہم اپنے الگورتھم یا فلوچارٹ میں کچھ غلط لکھتے ہیں:

(الف) منطقی غلطی (ب) رن ٹائم غلطی (ج) سینٹیکس غلطی (د) لوکل غلطی

-18 یہ اس وقت ہوتی ہیں جب الگورتھم یا فلوچارٹ پر عمل درآمد کیا جا رہا ہوتا ہے:

(الف) منطقی غلطی (ب) رن ٹائم غلطی (ج) سینٹیکس غلطی (د) لوکل غلطی

-19 فیصلے کے مرحلے پر غلط سمبلو کا استعمال کرنا مثال ہے:

(الف) منطقی غلطی کی (ب) رن ٹائم غلطی کی (ج) سینٹیکس غلطی کی (د) لوکل غلطی کی

-20 اس کی غلطیوں کو تلاش کرنا سب سے آسان ہے کیونکہ LARP ٹول عام طور پر ان کی نشاندہی کرتا ہے:

(الف) منطقی (ب) رن ٹائم (ج) سینٹیکس (د) لوکل

-21 کسی مسئلے کو سمجھنے میں شامل ہے:

(الف) بنیادی مسئلے کی شناخت کرنا (ب) ضروریات کی وضاحت کرنا (ج) مقاصد کا تعین کرنا (د) یہ تمام

www.ilmkidunya.com

22- یہ عمل میں ایک فیصلہ کی جگہ کی نمائندگی کرتا ہے جہاں بہاؤ ہاں یا نہیں، سوال صحیح یا غلط کی بنیاد پر مختلف ہو سکتا ہے:
(الف) ڈائمنڈ (ب) تیر (ج) مستطیل (د) بیضوی

23- فلوچارٹ کو کمپیوٹر سائنس دان جیسے کہ _____ نے کمپیوٹنگ کے ابتدائی دنوں میں مقبول بنایا:
(الف) جان وان نیومین (ب) ہرمن گولڈسٹائن (ج) الف اور ب دونوں (د) کوئی بھی نہیں

24- اصل کوڈ لکھنے سے پہلے سافٹ ویئر کی تیاری میں _____ کو اکثر استعمال کیا جاتا ہے تاکہ اس بات کو یقینی بنایا جا سکے کہ منطق درست ہے:
(الف) فلوچارٹ (ب) سوڈوکوڈ (ج) ہارڈ ویئر (د) انٹرنل سسٹم

25- ان غلطیوں کو تلاش کرنا سب سے مشکل ہے کیونکہ الگورتھم اب بھی چلتا ہے لیکن صحیح جوابات نہیں دیتا:
(الف) منطقی (ب) رن ٹائم (ج) سینٹیکس (د) لوکل

جوابات

1-	(ج)	2-	(ب)	3-	(الف)	4-	(ب)	5-	(الف)
6-	(د)	7-	(ب)	8-	(ج)	9-	(الف)	10-	(الف)
11-	(الف)	12-	(ب)	13-	(د)	14-	(الف)	15-	(ب)
16-	(د)	17-	(ج)	18-	(ب)	19-	(الف)	20-	(ج)
21-	(د)	22-	(الف)	23-	(ج)	24-	(ب)	25-	(الف)

اضافی مختصر سوالات (بمطابق حاصلاتِ تعلّم)

✪ دیے گئے سوالات کے مختصر جواب تحریر کریں۔

1- کمپیوٹیشنل ٹھنکنگ کیا 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 کی تعریف کریں۔

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

3- پیٹرن کی شناخت سے کیا مراد ہے؟

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

4- تجرید کے بارے میں آپ کیا جانتے ہیں؟

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 کی تعریف کریں۔

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

www.ilmkidunya.com

6۔ مسئلے کو سمجھنا کیوں ضروری 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 مسئلے کو آسان بنانے سے کیا مراد ہے؟

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

8۔ بہترین حل کے انتخاب سے کیا مراد ہے؟

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

9۔ فلوچارٹ پر روشنی ڈالیں۔

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

10۔ سوڈوکوڈ پر مختصر نوٹ تحریر کریں۔

جواب: سوڈوکوڈ 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 کے بارے میں آپ کیا جانتے ہیں؟

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

12۔ سپیس کمپلیکسٹی کے کام کا مختصراً جائزہ لیں۔

جواب: سپیس کمپلیکسٹی ان پٹ سائز کے سلسلے میں الگورتھم 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 کس کام آتا 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 کب ہوتا 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 کا کیا مطلب 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 کے کردار پر روشنی ڈالیں۔

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

17۔ کیا کمپیوٹیشنل تھنکنگ صرف کمپیوٹر سائنس تک محدود ہے؟

جواب: 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

www.ilmkidunya.com