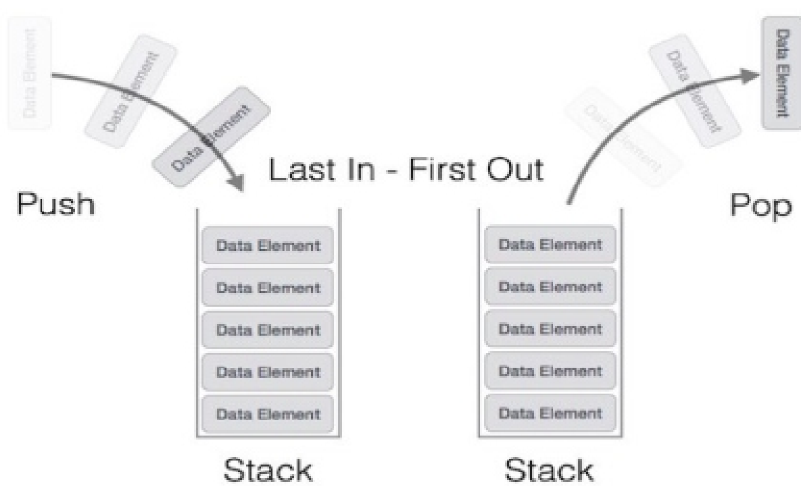
अध्याय 4

स्टैक और क्यु

स्टैक(Stack): स्टैक एक एब्सट्रैक्ट डाटा टाइप (एडीटी) है जिसका प्रयोग आमतौर पर सभी प्रोग्रामिंग भाषाओं में किया जाता है। उदाहरण के लिए–ताश के पत्तों की स्टैक या एक डेक या प्लेटों की स्टैक वास्तविक दुनिया में एक स्टैक की तरह बर्ताव करती है

वास्तविक दुनिया में स्टैक केवल एक छोर पर ऑपरेशन की अनुमति देती है। उदाहरण के लिए– हम केवल कार्ड या प्लेट स्टैक के ऊपर से तत्व रख या निकाल सकते हैं। इसी तरह, स्टैक एडीटी केवल एक छोर पर डेटा के ऑपरेशन की अनुमति देता है। किसी भी समय, हम केवल स्टैक के शीर्ष तत्व का उपयोग कर सकते हैं। यह सुविधा स्टैक को LIFO(अंतिम–इन–फर्स्ट–आउट) डेटा स्ट्रक्चरबनाता है। यहाँ पर जो तत्व अंत में जोड़ा जाता है पहले हटाया जाता है स्टैक शब्दावली में जोड़ने को पुश (Push) तथा हटाने को पॉप (Pop) आपरेशन कहा जाता है।

स्टैक प्रेजेंटेशन (Presentation): निम्नलिखित चित्र मेंएक स्टैक और इसके ऑपरेशन को दर्शाया गया है –



एक स्टैक को ऐरे, स्ट्रक्चर, पॉइंटर और लिंक्ड लिस्ट के माध्यम से इम्पलीमेंट किया जा सकता है। एक स्टैक फिक्स साइज या डायनामिक साइज दोनों में से एक प्रकार का हो सकता है। हम यहाँ एक ऐरे का उपयोग कर स्टैक को इम्पलीमेंट कर रहे है जो एक फिक्स साइज स्टैक है।

बुनियादी ऑपरेशन:स्टैक ऑपरेशन का उपयोग स्टैक को इनिशलायजिंग और डीइनिशलायजिंग करने के लिए किया जाता है। इसके अलावा एक स्टैक निम्नलिखित दो प्राथमिक कार्यों के लिए प्रयोग किया जाता है –

Push() –एक तत्व स्टैक में जोड़ना

Pop()– एक तत्व स्टैक से हटाना

जब हम स्टैक में कोई तत्व जोड़ते है तब स्टैक के कुशलता से उपयोग के लिए उसका स्टेटस चेक करते है इसके लिए निम्नलिखित फंक्शन का उपयोग करते है–

peak() – स्टैक के शीर्ष डेटा तत्व को हटाये बिना प्राप्त करना

isFull() –स्टैक के भरे होने की जाच करना

isEmpty() – स्टैक के खाली होने की जाच करना

हर समय स्टैक में अंतिम जोड़े गए तत्व के लिये एक पॉइंटर का उपयोग करते है जो की स्टैक के टॉप को बताता है इसे TOP के नाम से जाना जाता है टॉप वेरिएबल बिना हटाये स्टैक के टॉप की वैल्यू बताता है

स्टैक फंक्शन के प्रोसीजर –

peek():

peek() फंक्शन के लिए एल्गोरिथ्म–

begin procedure peek
return stack[top]
end procedure

peek() फंक्शन का **C** भाषा में इम्प्लीमेंटेशन –

```
int peek() {
return stack[top];
}
```

isfull():

isfull() फंक्शन के लिए एल्गोरिथ्म–

begin procedure isfull
if top equals to MAXSIZE
return true

```
else
return false
endif
end procedure
```

isfull() फंक्शन का **C** भाषा में इम्प्लीमेंटेशन –

```
bool isfull() {
if(top == MAXSIZE)
return true;
else
return false;
}
```

isempty():

isempty() फंक्शन के लिए एल्गोरिथ्म –

```
begin procedure isempty
if top less than 1
return true
else
return false
endif
end procedure
```

isempty()फंक्शन का C भाषा में इम्प्लीमेंटेशन–

Example

```
bool isempty() {
if(top == -1)
return true;
else
return false;
}
```

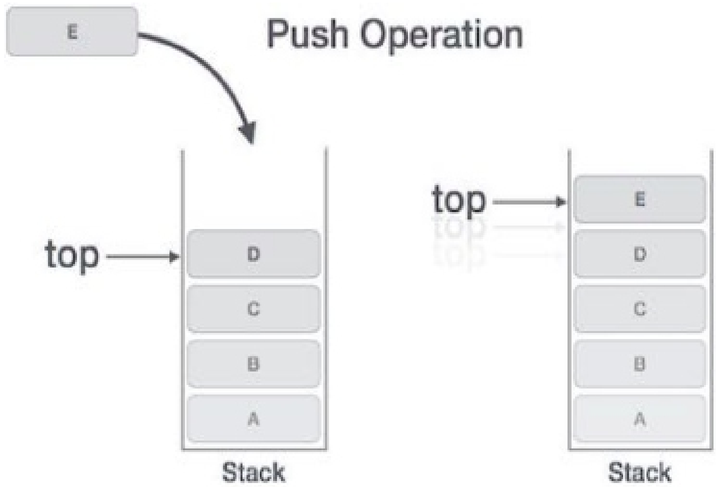
पुश ऑपरेशन(Push Operation):स्टैक में एक नया डेटा तत्व जोड़ने या डालने की प्रक्रिया को पुश आपरेशन कहते है पुश आपरेशन निम्नलिखित स्टेप्स की एक श्रृंखला है।

Step 1 – Checks if the stack is full.

Step 2 – If the stack is full, produces an error and exit.

Step 3 – If the stack is not full, increments top to point next empty space.

Step 4 – Adds data element to the stack location, where top is pointing.



Step 5 – Returns success.

यदि स्टैक को लिंक लिस्ट से इम्पलीमेंट करते है तो स्टेप 3 में डायनामिक मेमोरी आवंटित करनी होगी।

पुश आपरेशन के लिए एल्गोरिथ्म–

begin procedure push: stack, data

if stack is full

return null

endif

top ← top + 1

stack[top] ← data

end procedure

एल्गोरिथ्म का C में इम्प्लीमेंटेशन –

void push(int data) {

if(!isFull()) {

top = top + 1;

stack[top] = data;

} else {

printf("Could not insert data, Stack is full.\n");

}
}

पॉप आपरेशन(Pop Operation): स्टैक से एक डेटा तत्व को हटाने की प्रक्रिया को पॉप आपरेशन कहते है जब पॉप आपरेशन को एक ऐरे की मदद से इम्पलीमेंट करते है तो वास्तव में डाटा तत्व को हटाने की बजाए टॉप वेरिएबल को एक से घटाते है जबकि लिंक लिस्ट से इम्पलीमेंट करने पर वास्तव में डाटा तत्व को हटा कर मेमोरी को deallocates किया जाता है पॉप आपरेशन निम्नलिखित स्टेप्स की एक श्रृंखला है –

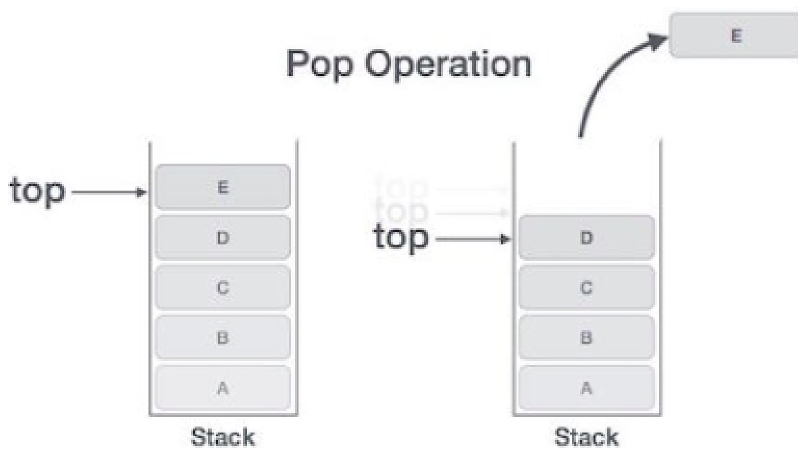
Step 1 – Checks if the stack is empty.

Step 2 – If the stack is empty, produces an error and exit.

Step 3 – If the stack is not empty, accesses the data element at which top is pointing.

Step 4 – Decreases the value of top by 1.

Step 5 – Returns success.



पॉपआपरेशन के लिए एल्गोरिथ्म–

begin procedure pop: stack

if stack is empty

return null

endif

data ← stack[top]

top ← top - 1

return data

end procedure

एल्गोरिथ्म का C में इम्प्लीमेंटेशन –

```
int pop(int data) {
if(!isempty()) {
data = stack[top];
top = top - 1;
return data;
} else {
printf("Could not retrieve data, Stack is empty.\n");
}
}
```

स्टैक के उपयोग–निम्नलिखित कार्यो के लिये स्टैक का उपयोग किया जा सकता है।

(क) अंकगणित अभिव्यक्ति मूल्यांकन(अरिथमैटिक एक्सप्रेशन इवैल्यूएशन)

(ख) बकट्रैकिंग

(ग) स्मृति प्रबंधन(मेमोरी मैनेजमेंट)

(क) अंकगणित अभिव्यक्ति मूल्यांकन(अरिथमैटिक एक्सप्रेशन इवैल्यूएशन): अरिथमैटिक एक्सप्रेशन लिखने के तरिके को नोटेशन कहते है एक अरिथमैटिक एक्सप्रेशन को तीन अलग अलग तरीको लेकिन समान नोटेशन में लिख सकते है बिना सार या आउटपुट के बदले। निम्नलिखित तीन नोटेशन है–

इन्फिक्स नोटेशन

उपसर्ग (पोलिश) नोटेशन

पोस्टफिक्स (रिवर्स पोलिश) नोटेशन

इन एक्सप्रेशन का नाम ऑपरेटर के उपयोग के अनुसार दिया गया है।

इन्फिक्स नोटेशन

हम एक एक्सप्रेशन a - b + c लिखते है जिसमे में ऑपरेटर ऑपरेंड के मध्य इस्तेमाल किया गया है ये एक इन्फिक्स नोटेशन है इसका मनुष्य के लिये पढना,लिखना और बोलना आसान है लेकिन कंप्यूटिंग उपकरणों के लिए मुश्किल है एक एल्गोरिथ्म में इन्फिक्स एक्सप्रेशन को प्रोसेस करने लिए अधिक टाइम और स्पेस की आवश्यकता होती है।

उपसर्ग (पोलिश) नोटेशन

इस नोटेशन में ऑपरेटर ऑपरेंड के आगे लिखा होता है उदाहरण के लिए +ab जो की इन्फिक्स नोटेशन a + b के समान है उपसर्ग नोटेशन को पोलिश नोटेशन भी कहते है।

पोस्टफिक्स नोटेशन

पोस्टफिक्स नोटेशन को रिवर्स पोलिश नोटेशन कहते है इसमें ऑपरेटर ऑपरेंड के बाद में होता है उदाहरण के लिए ab+ जो की इन्फिक्स नोटेशन a + b के समान है।

स्टैक का उपयोग एक नोटेशन को दूसरे नोटेशन में रूपांतरण के लिए किया जाता है।

(ख) बकट्रैकिंग:बकट्रैकिंग का प्रयोग एल्गोरिथ्म में किया जाता है जहां किसी पथ के साथ स्टेप्स होते है जो किसी स्टार्ट पॉइंट से किसी उद्देश्य तक हो। उदाहरण के लिए

एक भूलभुलैया के माध्यम से अपना रास्ता सर्च करना।

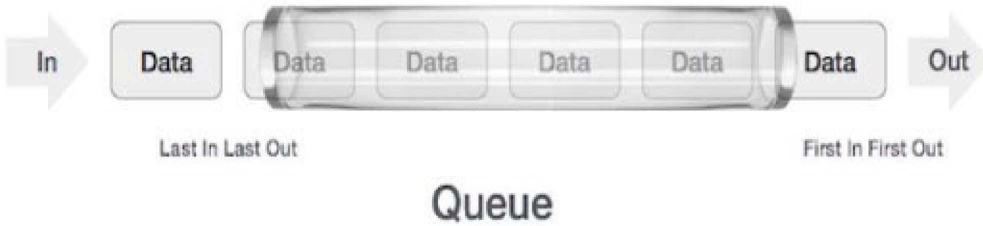
एक ग्राफ में एक पॉइंट से दूसरे पॉइंट तक रास्ता पता करना।

उपरोक्त सभी मामलो में एक पॉइंट से दुसरे पॉइंट तक जाने के लिए बहुत सारे विकल्प होते है यदि एक पॉइंट से दुसरे पॉइंट पर जाने के बाद वापस पहले पॉइंट पर आना हो और अन्य विकप्ल चुनना हो।
फिर, समाधान के लिए स्टैक का इस्तेमाल किया जा सकता है। रिकर्शन एक अन्य ईस्ट समाधान है जिसको स्टैक की मदद से इम्पलीमेंट कर सकते है।

(ग) स्मृति प्रबंधन: कोई भी आधुनिक कंप्यूटर अपने प्रोग्राम को रन करने के लिए प्राथमिक स्मृति प्रबंधन मॉडल के रूप में एक स्टैक उपयोग करता है।

क्यु (Queue):क्यु एक एब्सट्रेक्ट डेटा स्ट्रक्चर है जो कुछ हद तक स्टैक के समान है। स्टैक के विपरीत, एक क्यु अपने दोनों शीरों पर खुला होता है। एक शीरा तत्व को जोड़ने (enqueue) और दूसरा शीरा तत्व को हटाने(dequeue) के उपयोग में आता है क्यु डेटा स्ट्रक्चर पहले आओ पहले जाओ (FIFO) के सिद्धांत पर काम करता है अर्थात, पहले संग्रहीत डेटा आइटम पहले हटा दिए जायंगे।

क्यु के लिए एक वास्तविक दुनिया में एक सिंगल लेन का रोड जिसमे जो साधन पहले प्रवेश करेगा पहले बाहर आएगा एक उदाहरण के रूप में देखा जा सकता है।



क्यु प्रेजेंटेशन (Presentation):अब हम समझ गए की क्यु में दोनों सिरो को अलग अलग कारण के लिए उपयोग में लेते है निम्नलिखित चित्र की सहायता से क्यु का डाटा स्ट्रक्चर के रूम में प्रेजेंटेशन है।

एक स्टैक की तरह क्यु को भी ऐरे, स्ट्रक्चर, पॉइंटर और लिंक्ड लिस्ट के माध्यम से इम्पलीमेंट किया जा सकता है। एक क्यु फिक्स साइज या डायनामिक साइज दोनों में से एक प्रकार का हो सकता है। हम यहाँ एक ऐरे का उपयोग कर क्यु को इम्पलीमेंट कर रहे है जो एक फिक्स साइज क्यु है।

क्यु के बुनियादी ऑपरेशन: क्युऑपरेशन का उपयोग क्यु को इनिशलायजिंग और डीइनिशलायजिंग करने के लिए किया जाता है।निम्नलिखित क्यु के बुनियादी आपरेशन है।

enqueue() – एक तत्व क्यु में जोड़ना

dequeue() – एक तत्व क्यु से हटाना

क्यु के कुशलता से उपयोग के लिए निम्नलिखित फंक्शनस का उपयोग करते है–

peek() – क्यु के शीर्ष डेटा तत्व को हटाये बिना प्राप्त करना

isfull() – क्यु के भरे होने की जाच करना

isempty() – क्यु के खाली होने की जाच करना

क्यु के सपोर्टिव फंक्शनस निम्न है।

peek()

peek() फंक्शन के लिए एल्गोरिथ्म–

begin procedure peek

return queue[front]

end procedure

peek() फंक्शन का **C** भाषा में इम्प्लीमेंटेशन –

Example

int peek() {

return queue[front];

}

isfull():

यहाँ पर हम क्यु को इम्पलीमेंट करने के लिए एक आयामी ऐरे का उपयोग कर रहे है क्यु के भरे होने की जाच के लिए हम केवल रियर पॉइंटर को चेक करते है की वह मैक्स साइज के बराबर है या नहीं। यदि हम क्यु को सर्कुलर लिंक लिस्ट से इम्पलीमेंट करते है तो अलग एल्गोरिथ्म होगी।

isfull( ) फंक्शन के लिए एल्गोरिथ्म–

```
begin procedure isfull
if rear equals to MAXSIZE
return true
else
return false
endif
end procedure
```

isfull( )फंक्शन का **C** भाषा में इम्प्लीमेंटेशन –

```
bool isfull() {
if(rear == MAXSIZE - 1)
return true;
else
return false;
}
```

isempty():

isempty() फंक्शन के लिए एल्गोरिथ्म–

```
begin procedure isempty
if front is less than MIN  OR front is greater than rear
return true
else
return false
endif
end procedure
```

यदि फ्रंट का मान मिन या 0 से कम है तो इसका मतलब क्यु को इनिसलाएज नहीं किया है और क्यु खाली है।

isempty() फंक्शन का C भाषा में इम्प्लीमेंटेशन –

```
bool isempty() {
if(front < 0 || front > rear)
return true;
else
return false;
```

एनक्यु (Enqueue) आपरेशन: क्यु में दो डाटा पॉइंटर फ्रंट और रियर होते है इसलिए इसके आपरेशन स्टैक से कठिन होते है क्यु में डाटा तत्व को जोड़ने (Insert) के लिए निम्नलिखित स्टेप्स का उपयोग करते हैं।

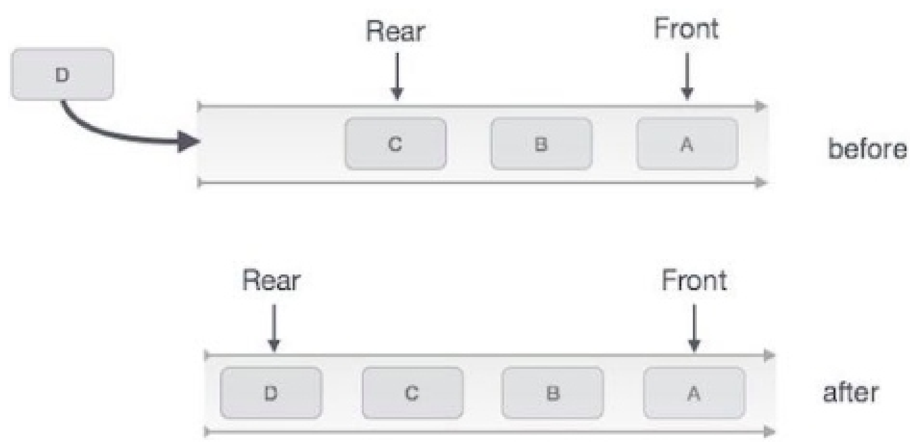
Step 1 – Check if the queue is full.

Step 2 – If the queue is full, produce overflow error and exit.

Step 3 – If the queue is not full, increment rear pointer to point the next empty space.

Step 4 – Add data element to the queue location, where the rear is pointing.

Step 5 – return success.



Queue Enqueue

एनक्यु (Enqueue) आपरेशन के लिए एल्गोरिथ्म–

procedure enqueue(data)

if queue is full

return overflow

endif

rear ← rear + 1

queue[rear] ← data

return true

end procedure

एनक्यु (Enqueue) आपरेशन का C भाषा में इम्प्लीमेंटेशन –

```
int enqueue(int data)
if(isfull())
return 0;
rear = rear + 1;
queue[rear] = data;
return 1;
end procedure
```

डीक्यु (Dequeue) आपरेशन: डाटा तत्व को क्यु से हटाने का काम दो भागो में किया जाता है एक उस डाटा तत्व को एक्सेस करना जहा पॉइंटर पॉइंट कर रहा हो और दूसरा उसको वहाँ से हटाना। निम्नलिखित स्टेप्स से डीक्युआपरेशन परफॉर्म किया जाता है–

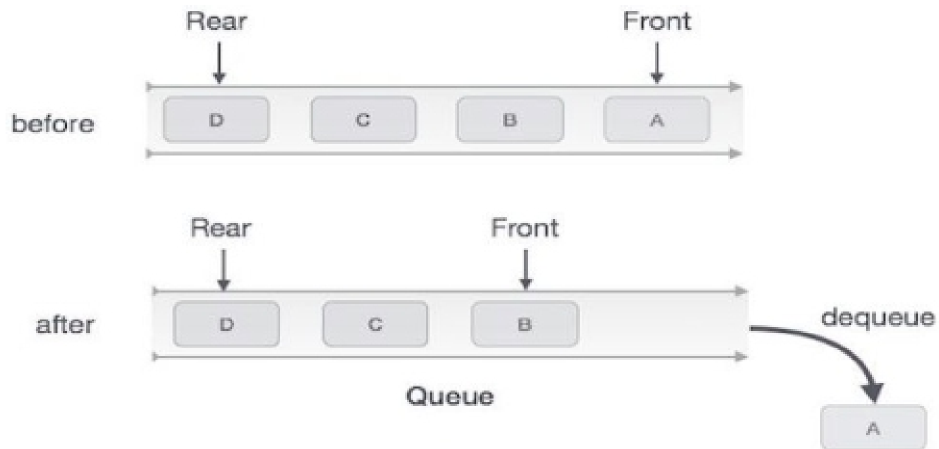
Step 1 – Check if the queue is empty.

Step 2 – If the queue is empty, produce underflow error and exit.

Step 3 – If the queue is not empty, access the data where front is pointing.

Step 3 – Increment front pointer to point to the next available data element.

Step 5 – Return success.



Queue Dequeue

डीक्यु (Dequeue) आपरेशन के लिए एल्गोरिथ्म–

procedure dequeue

if queue is empty

return underflow

end if

data = queue[front]

front ← front + 1

return true

end procedure

डीक्यु (Dequeue) आपरेशन का C भाषा में इम्प्लीमेंटेशन –

```
int dequeue() {
if(isempty())
return 0;
int data = queue[front];
front = front + 1;
return data;
}
```

महत्वपूर्ण बिंदु

- स्टैक एक एब्सट्रैक्ट डाटा टाइप (एडीटी) है जिसका प्रयोग आमतौर पर सभी प्रोग्रामिंग भाषाओं में किया जाता है।
- एक स्टैक को ऐरे, स्ट्रक्चर, पॉइंटर और लिंक्ड लिस्ट के माध्यम से इम्पलीमेंट किया जा सकता है। एक स्टैक फिक्स साइज या डायनामिक साइज दोनों में से एक प्रकार का हो सकता है।
- क्यु एक एब्सट्रेक्ट डेटा स्ट्रक्चर है जो कुछ हद तक स्टैक के समान है। स्टैक के विपरीत, एक क्यु अपने दोनों शीरों पर खुला होता है।
- कोई भी आधुनिक कंप्यूटर अपने प्रोग्राम को रन करने के लिए प्राथमिक स्मृति प्रबंधन मॉडल के रूप में एक स्टैक उपयोग करता है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न

प्र1 निम्न में से कौन सा नाम स्टैक से संबंधित नहीं है।

(अ)FIFO सूची (ब)LIFO सूची

(स)POP (द)Push

प्र2 शब्द Push और POP किस से संबंधित है।

(अ) ऐरे (ब) लिस्ट

(स) स्टैक (द) ऊपर के सभी

प्र3 एक डेटा स्ट्रक्चर जहां तत्वों का जोड़ना या हटाना किसी भी सिरे पर किया जा सकता है लेकिन बीच में नहीं।

(अ) लिंक लिस्ट (ब) स्टैक

(स) क्यु (द) डीक्यु

प्र4 ग्राफ में Breadth First Traversal के लिए आवश्यक डेटा स्ट्रक्चर है।

(अ) स्टैक (ब) ऐरे

(स) क्यु (द) टी(Tree)

प्र5 एक क्यु है।

(अ)FIFO लिस्ट (ब)LIFO लिस्ट

(स) ओर्डड ऐरे (द) रैखिक Tree

लघुत्तरात्मक प्रश्न

प्र1 स्टैक को परिभाषित करें।

प्र2 क्यु को परिभाषित करें।

प्र3 पुश ऑपरेशन क्या है?

प्र4 पॉप ऑपरेशन क्या है?

निबंधात्मक प्रश्न

प्र1 स्टैक डेटा स्ट्रक्चर के एपलीकेशन को समझाओं।

प्र2 स्टैक ऑपरेशन को विस्तार से समझाओ।

प्र3 विस्तार में सरक्युलर क्यु को समझाओ।

प्र4 डीक्यु को समझाओ।

उत्तरमाला

उत्तर 1: अउत्तर 2: स उत्तर 3: द

उत्तर 4: स उत्तर 5: अ