23MX128

Lt Col T V Perumal(Retd)

**Colliding strategy**

Declare a single array of size N elements

Divide the array into 2 logical partitions to represent the two stacks

Name the two stacks as LeftStack and RightStack

LeftStack gets filled from the left side of the array

RightStack gets filled from the right of the array

Declare variables as under

LeftStackTop – indicates the index of the top most element in LeftStack

RightStackTop – indicates the index of the top most element in RightStack

LeftStackSize – the number of elements in LeftStack

RightStackSize – the number of elements in RightStack

Initialize variables as under

LeftStackTop = -1; LeftStackSize = 0;

RightStackTop = N; RightStackSize = 0;

1. To push an element value X into stacks :-

if LeftStackSize + RightStackSize = N then print “Stack Full” return

Else If LeftStackTop < (RightStackTop -1) // push into left stack

Push into LeftStack as under :-

{ LeftStackTop +=1

LeftStack[LeftStackTop] = X

LeftStackSize +=1

}

Else // push into right stack

Push into RighttStack as under :-

{ RightStackTop -=1

RightStack[RightStackTop] = X

RightStackSize +=1

}

2. To pop elements from respective stacks

Function Pop (Stack)

{ If Stack passed is LeftStack

{ if LeftStackTop = -1 print “Stack Empty” return

Else { print “Element popped : “ + LeftStack[LeftStackTop]

LeftStackTop -=1 ;

LeftStackSize -=1 ;

}

}

Else // Stack passed is RightStack

{ if RightStackTop = N print “Stack Empty” return

Else { print “Element popped : “ + RightStack[RightStackTop]

RightStackTop +-=1 ;

RightStackSize -=1 ;

}

}

}