# STACKS-2

## Solutions

**1.** The result evaluating the postfix expression 10 5 + 60 6 / * 8 – is

(a) 284 (b) 213
(c) 142 (d) 71

**Solution:** Option ( c )

**2.** Which one of the following is an application of Stack Data Structure?

(a) Managing function calls (b) The stock span problem
(c) Arithmetic expression evaluation (d) All of the above

**Solution:** Option (d)

**3.** Which of the following is true about linked list implementation of stack?

(a) In push operation, if new nodes are inserted at the beginning of linked list, then in pop operation, nodes must be removed from end.

(b) In push operation, if new nodes are inserted at the end, then in pop operation, nodes must be removed from the beginning.

(c) Both of the above.

(d) None of the above.

**Solution:** Option (d)

**Explanation:**
To keep the Last In First Out order, a stack can be implemented using linked list in two ways:

a) In push operation, if new nodes are inserted at the beginning of linked list, then in pop operation, nodes must be removed from beginning.

b) In push operation, if new nodes are inserted at the end of linked list, then in pop operation, nodes must be removed from end.

**4.** Consider the following pseudo code that uses a stack:

```
declare a stack of characters

while ( there are more characters in the word to read )
{
  read a character
  push the character on the stack
}

while ( the stack is not empty )
{
  pop a character off the stack
  write the character to the screen
}
```

What is output for input “ravindras”?

(a) ravindrasravindras
(b) sardnivar
(c) ravindras
(d) sardnivarsardnivar

**Solution:** Option (b)

**Explanation:**
Since the stack data structure follows LIFO order. When we pop() items from stack, they are popped in reverse order of their insertion (or push()).

**5.** Following is an incorrect pseudo code for the algorithm which is supposed to determine whether a sequence of parentheses is balanced:

```
declare a character stack

while ( more input is available )
{
  read a character
  if ( the character is a '(' )
    push it on the stack
  else if ( the character is a ')' and the stack is not empty )
    pop a character off the stack
  else
    print "unbalanced" and exit
```

}
print "balanced"

Which of these unbalanced sequences does the above code think is balanced?

(a) ((())
(b) ())(()
(c) (()()))
(d) (()))()

**Solution:** Option (a)

**Explanation:**
At the end of while loop, we must check whether the stack is empty or not. For input ((()), the stack doesn't remain empty after the loop.

**6.** The following postfix expression with single digit operands is evaluated using a stack:

8 2 3 ^ / 2 3 * + 5 1 * -

Note that ^ is the exponentiation operator. The top two elements of the stack after the first * is evaluated are:

(a) 6, 1
(b) 5, 7
(c) 3, 2
(d) 1, 5

**Solution:** Option (a)

**7.** Let S be a stack of size n >= 1. Starting with the empty stack, suppose we push the first n natural numbers in sequence, and then perform n pop operations. Assume that Push and Pop operation take X seconds each, and Y seconds elapse between the end of one such stack operation and the start of the next operation. For m >= 1, define the stack-life of m as the time elapsed from the end of Push(m) to the start of the pop operation that removes m from S. The average stack-life of an element of this stack is

(a) n(X+ Y)
(b) 3Y + 2X
(c) n(X + Y)-X
(d) Y + 2X

**Solution:** Option (c)

**Explanation:**
We can easily arrive at the result by taking few examples.

**8.** A single array A[1..MAXSIZE] is used to implement two stacks. The two stacks grow from opposite ends of the array. Variables top1 and top2 (topl < top 2) point to the location of the topmost element in each of the stacks. If the space is to be used efficiently, the condition for "stack full" is (GATE CS 2004)

(a) (top1 = MAXSIZE/2) and (top2 = MAXSIZE/2+1)
(b) top1 + top2 = MAXSIZE
(c) (top1= MAXSIZE/2) or (top2 = MAXSIZE)
(d) top1= top2 -1

**Solution:** Option (d)

**Explanation:**
If we are to use space efficiently then size of the any stack can be more than MAXSIZE/2.

Both stacks will grow from both ends and if any of the stack top reaches near to the other top then stacks are full. So the condition will be top1 = top2 -1 (given that top1 < top2).

**9.** Assume that the operators +, -, × are left associative and ^ is right associative. The order of precedence (from highest to lowest) is ^, x , +, -. The postfix expression corresponding to the infix expression a + b × c – d ^ e ^ f is

(a) abc × + def ^ ^ –
(b) abc × + de ^ f ^ –
(c) ab + c × d – e ^ f ^
(d) – + a × bc ^ ^ def

**Solution:** Option (a)

**10.** To evaluate an expression without any embedded function calls:

(a) One stack is enough
(b) Two stacks are needed
(c) As many stacks as the height of the expression tree are needed
(d) A Turing machine is needed in the general case

**Solution:** Option (a)

**11.** Following is C like pseudo code of a function that takes a number as an argument, and uses a stack S to do processing.

```
void fun(int n)
{
   Stack S;  // Say it creates an empty stack S
   while (n > 0)
   {
     // This line pushes the value of n%2 to stack S
     push(&S, n%2);
     n = n/2;
     }
   // Run while Stack S is not empty
   while (!isEmpty(&S))
     printf("%d ", pop(&S)); // pop an element from S and print it
}
```

What does the above function do in general?

(a) Prints binary representation of n in reverse order
(b) Prints binary representation of n
(c) Prints the value of Logn
(d) Prints the value of Logn in reverse order

**Solution:** Option (b)

**12.** In linked representation of stack ..... holds the elements of the stack.

(a) INFO fields
(b) TOP fields
(c) LINK fields
(d) NULL fields

**Solution:** Option (a)

**13.** ..... form of access is used to add remove nodes from a stack.

(a) LIFO
(b) FIFO
(c) Both A and B
(d) None of these

**Solution:** Option (a)

**14.** In the linked representation of the stack ..... behaves as the top pointer variable of stack.

(a) Stop pointer
(b) Begin pointer
(c) Start pointer
(d) Avail pointer

**Solution:** Option (c)

**15.** Entries in a stack are "ordered". What is the meaning of this statement?

(a) A collection of stacks can be sorted.
(b) Stack entries may be compared with the '< ' operation.
(c) The entries must be stored in a linked list.
(d) There is a first entry, a second entry, and so on.

**Solution:** Option (d)

**16.** The operation for adding an entry to a stack is traditionally called:

(a) add
(b) append
(c) insert
(d) push

**Solution:** Option (d)

**17.** The operation for removing an entry from a stack is traditionally called:

(a) delete
(b) peek
(c) pop
(d) remove

**Solution:** Option (c)

**18.** Which of the following stack operations could result in stack underflow?

(a) is_empty
(b) pop
(c) push
(d) Two or more of the above answers

**Solution:** Option (b)

**19.** Which of the following applications may use a stack?

(a) A parentheses balancing program.
(b) Keeping track of local variables at run time.
(c) Syntax analyzer for a compiler.
(d) All of the above.

**Solution:** Option (d)

**20.** Which of the following permutations can be obtained in the output(in the same order using the stack),assuming that the input is the sequence 1, 2, 3, 4, 5 in that order?

(a) 3, 4, 5, 1, 2
(b) 3, 4, 5, 2, 1
(c) 1, 5, 2, 3, 4
(d) 5, 4, 3, 2, 1

**Solution:** Option (b)

**Explanation:**

Input 1

Push 1 to stack ------- stack {1}

Input 2

Push 2 to stack -------stack{1,2}

Input 3

Push 3 to stack---------- stack {1,2,3}

Pop( ) ---------- stack{1,2}

Input 4

Push 4 to stack ---------- stack{1,2,4}

Pop() ---------- stack{1,2}

Input 5

Push 5 to stack ---------- stack{1,2,5}

Pop() ---------- stack{1,2}

Pop() ---------- stack{1}

Pop() ---------- stack{}

If the popped elements are printed then the output will match with that of option b