Data Structures Test Bank 2015

Data Structures Stack

Question 1

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

Question 1 Explanation:

See method 2 of http://www.geeksforgeeks.org/binary-representation-of-a-given-number/ for explanation.

Data Structures Test Bank 2015

Question 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

Question 2 Explanation:

See http://en.wikipedia.org/wiki/Stack_(abstract_data_type)#Applications

Question 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

Question 3 Explanation:

To keep the **L**ast **I**n **F**irst **O**ut 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.

Data Structures Test Bank 2015

Question 4

Consider the following pseudocode 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 "geeksquiz"?

A	geeksquizgeeksquiz
B	ziuqskeeg
C	geeksquiz
D	ziuqskeegziuqskeeg

Question 4 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())

Data Structures Test Bank 2015

Question 5

Following is an incorrect pseudocode 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? Source: http://www.cs.colorado.edu/~main/questions/chap07q.html

A (((())

B ())(()

C ((()))

D ((()))()

Question 5 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.

Data Structures Test Bank 2015

Question 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

Question 6 Explanation:

The algorithm for evaluating any postfix expression is fairly straightforward:

1. While there are input tokens left
   - o Read the next token from input.
   - o If the token is a value
     - + Push it onto the stack.
   - o Otherwise, the token is an operator

     (operator here includes both operators, and functions).
     - * It is known a priori that the operator takes n arguments.
     - * If there are fewer than n values on the stack

       (Error) The user has not input sufficient values in the expression.
     - * Else, Pop the top n values from the stack.
     - * Evaluate the operator, with the values as arguments.
     - * Push the returned results, if any, back onto the stack.
2. If there is only one value in the stack
   - o That value is the result of the calculation.
3. If there are more values in the stack
   - o **(Error)** The user input has too many values.

Source for algorithm: http://en.wikipedia.org/wiki/Reverse_Polish_notation#The_postfix_algorithm Let us run the above algorithm for the given expression. First three tokens are values, so they are simply pushed. After pushing 8, 2 and 3, the stack is as follows

8, 2, 3

When ^ is read, top two are popped and power(2^3) is calculated

8, 8

When / is read, top two are popped and division(8/8) is performed

1

Next two tokens are values, so they are simply pushed. After pushing 2 and 3, the stack is as follows

1, 2, 3

When * comes, top two are popped and multiplication is performed.

1, 6

Question 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

Question 7 Explanation:

We can easily arrive at the result by taking few examples

Data Structures Test Bank 2015

Question 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

Question 8 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)

Question 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

Question 9 Explanation:

^ is right associative

Data Structures Test Bank 2015

Question 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

Question 11

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

A	284
B	213
C	142
D	71

مع تحيات مبرمجون بلا حدود 2015

تامر أبو الزينات 0786358466

محمود عماد 0785780440