Answer Script

Question No. 01
1. Between array-based stack implementation and linked-list-based stack implementation, which is better when I need random access in the stack? Explain the reasons.
Answer No. 01
If i want to random access in a stack value, then array base stack implementation is better because Think in a linked list base stack where one node connects with another node or one node carries another or next or previous node address but that node can not contain after next five node address. Therefore, i want to access the index 4 node value then what have to do. I have to do liner travers it will take time if the stack size is so bigger, then it will take so much time. Now think array base stack in the array every cell stays sequentially so i know every index array if i know array's first cell address so it will take less time if the stack size is bigger and it time complexity if constant operation which is O(1) where linked list base stack random access time complexity is O(n). So, i can make the decision if i want to random access in stack, then array base stack implementation is so better.

Question No. 02
1. What is the time complexity of push, pop and top operations in a stack?
Answer No. 02
In stack push which add a new element in head or first index where a single operation so its time complexity is O(1). Time complexity of pop operation also constant so its time complexity O(1) because stack work LIFO method. Time complexity of top operation also constant we just return or check which value has first index so its a single operation and time complexity is O(1).

Question No. 03
1. Suppose you need a stack of characters, a stack of integers and a stack of real numbers. How will you implement this scenario using a single stack?
Answer No. 03
If i need a stack for multiple data types then i will implement a scenario that can handle all data type

variables and that is the Template class system.

first, i define a stack for Template data which means when i use the stack or any user use or call the

stack that time user will define which type of data he/her want to handle.

Template class pseudo-code: using array to implement the stack

```
#include<bits/stdc++.h>
using namespace std;
template <class T> // T means data type which provide from user
class Stack
{
   public:
   T *a;
   int arr_cap;
   int stack_size;

   Stack()
   {
       stack_size=0;

       arr_cap=1;

       a= new T[arr_cap];

   }

   void incriseStackSize()

   {

       T *temp = new T [arr_cap*2];

       for(int i=0;i<arr_cap;i++)

       temp[i]=a[i];


       swap(a,temp);

       delete[] temp;

       arr_cap *=2;

   }

   void push(T data)

   {

       stack_size +=1;

       if(stack_size>arr_cap)

       {

           incriseStackSize();
       }
       a[stack_size-1]=data;
   }
   void pop()
   {
       if(stack_size==0)
       {
           cout<<"stack is Empty";
           return;
       }
       //a[stack_size-1]=0;

       stack_size--;
   }

   T top()
   {
       if(stack_size==0)
       {
           cout<<"Stack Is Empty"<<endl;
```

```
       T ob;

       return ob;

     }

     return a[stack_size-1];

   }

   int StackSize()

   {

     return stack_size;

   }


};

int main()

{

  Stack<int>stk;

  for(int i=0;i<5;i++)

  {

   //stk.push(rand()%10);

   stk.push(i+1);

  }

  cout<<stk.top()<<endl;
  stk.pop();

  cout<<stk.top()<<endl;

   stk.pop();

  cout<<stk.top()<<endl;

   stk.pop();

  cout<<stk.top()<<endl;

   stk.pop();

  cout<<stk.top()<<endl;

   stk.pop();

  cout<<stk.top()<<endl;




  Stack<char>stk2;

  stk2.push('a');

  stk2.push('b');

  cout<<stk2.top()<<endl;

  stk2.pop();


}
```

Question No. 04

1. What is a postfix expression and why do we need it? How is it evaluated using a stack for the below example? You need to show all the steps.

abc*+de*+

Answer No. 04

The postfix expression is first two is operant and third one is operator example is: ab+ here 'a' and 'b' is operant and '+' is operator.

The computer cna not understand infix expressions like a+b*c+d is Computer can not understand what have to do

in this expression to get an answer if can it will take so much time. But this expression converts to postfix expression the Computer can easily solve the expression. And how to solve this the explanation is below

The Postfix expression is: abc*+de*+
push serially opratent until got a operator if got a operator and the pop_back last two data and did a opration and push the operation value in the stack and again got opreant then push and got operator the pop_back last tow tada and so on.....
the visual implementation is below:

Exp: abc*+de*+ $\rightarrow$

$t_1 = a * c$

$St \rightarrow a\boxed{b\ c}$ now got a oprator '*' so pop back_back last two data and do a operation and push_back the operation value in the stack

$t_2 = a + t_1$

$St \rightarrow a t_1$ $\rightarrow$ now got a oprator '+' so pop back_back last two data and do a operation and push_back the operation value in the stack

$t_3 = d * e$

$St \rightarrow t_2 d e$ now got a oprator '*' so pop back_back last two data and do a operation and push_back the operation value in the stack

$t_4 = t_2 + t_3$

$St \rightarrow t_2 t_3$ now got a oprator '+' so pop back_back last two data and do a operation and push_back the operation value in the stack

$St \rightarrow t_4$ Ans

Question No. 05

1. Simulate balanced parentheses check using stack for the below example. You need to show all the steps.

(([ ][ ]{ ( ) }))

Answer No. 05

The theory of Regulaer bracket Sequence Check is: We know bracket is character data. I use to Check Stack data structure to input bracket is valid or not.

Example for a valid bracket Sequence:

if i got an opening bracket then push the stack and i got a closing bracket then check stack top bracket if its matches with top bracket then pop from stack and so on. If stack top bracket and closing bracket is not match then make deception its not Regular bracket Sequence.

pop

Main stack: [3, 4, 6 , 2 ,5]

→ temp [5]   Main stack → [3.4 ,6, 2] pop

→ temp [5, push

now pop data 2 less than temp top data so temp top data will push until 2 get its right position so 5 push in main stack

2→ temp [2]   mains $\xrightarrow{5}$ [3.4,6,5] pop

5→ temp [2,5]   mains → [3, 4, 6] pop

6→ temp [2,5.6] pop pop   Main → [3 . 4] pop

4→ temp [2,4]   main $\xrightarrow{5,6}$ [3 , 6, 5]

5→ temp [2, 4. 5]   main → [3 , 6]

6→ temp [2, 4, 5, 6]   main → [3] pop

Question No. 06
1. Sort a stack of integers using another stack for the below example. You need to show all the steps.   Stack -> [3,4,6,2,5]
Answer No. 06
Stack sort main idea is first take a temporary stack and first pop data from main stack and push the temporary stack and again pop data from main stack and push the temporary stack but have to check if i push the data is its right position or not. if not the right position the pop data from the temporary stack and push it main stack and again try to push data in the temporary stack it. if that data now got right position then again pop data from main stack and push it temporary stack in right position. after push all data and its right position swap the stack like swap(main_stack, temporary_stack). below the all steps:

Main stack: [3, 4, 6, 2, 5] pop

→ temp [5]   Main stack → [3, 4, 6, 2] pop

→ temp [5, push

now pop data 2 less than temp top data so temp top data will push until 2 get its right position so 5 push in main stack

$\xrightarrow{2}$ temp [2]   mains $\xrightarrow{5}$ [3, 4, 6, 5] pop

$\xrightarrow{5}$ temp [2, 5]   mains → [3, 4, 6] pop

$\xrightarrow{6}$ temp [2, 5, 6] pop pop   Main → [3, 4] pop

$\xrightarrow{4}$ temp [2, 4]   main $\xrightarrow{5, 6}$ [3, 6, 5]

$\xrightarrow{5}$ temp [2, 4, 5]   main → [3, 6]

$\xrightarrow{6}$ temp [2, 4, 5, 6] pop   main → [3]

$\xrightarrow{3}$ temp [2 . 3]   main $\xrightarrow{4\ 5 . 6}$ [6 , 5 , 4]

$\xrightarrow{4}$ temp [2 , 3 , 4]   main → [6 , 5]

$\xrightarrow{5}$ temp [2 , 3 , 4 , 5] main [6]

$\xrightarrow{6}$ temp [2 . 3 . 4 , 5 . 6] main [/]

now swap the stack swap(temp, main_satack)

Question No. 07

1. Convert the infix expression to postfix expression using a stack. You need to show all the steps.

a+b*c+d*e

Answer No. 07

If i want to convert infix expression to postfix expression then i use stack theory.
first i take a string where contains final output result another one is stack which contains operator

firstly if i got operaent the its add to result string. If got operator first check stack is empty
then push the operator if not empty thn check oprator precedence if stack top oprator precedence is
high then pop oprator and push resut string until getting less than or equal operator in stack.
after add all data in resut string then chack stack if in stack any operator stay in stack than add
with the result until empty stack. after that i got the final out-put and all steps simulation below:

a+b*c+d*e

1. result = " "

2. stack = [ ]

→ result $\xrightarrow{a}$ "a" stack → [ ]

$\xrightarrow{+}$ result → "a" stack → [+]

→ result → "ab" stack → [+ *]

→ result → "abc*+" stack → [+]

→ result → "abc*+d" stack → [+]

→ result → "abc*+d" stack → [+ *]

→ result → "abc*+de" → stack → [+ *]

→ result → "abc*+de*+" stack [ ]

→

Output