structure	structure.
• Easier to implement	• Slightly more complex to implement.
• Linear queues are suitable in situations where elements are inserted and removed strictly from one end.	• Circular queues are useful in scenarios where the process of insertion and removal wraps around.

Tutorials : 7

4/06/2023

Recap -

① Why are stacks useful?

to call functions and execution, to evaluate expressions, to undo/redo operations, to manage memory, to browse history, to backtrack algorithms etc.

② Reverse a string using stack?

```
# include <stdio.h>
# include <string.h>

# define   MAX_LENGTH 100

// Function to reverse a String using a stack
Void reverseString (char * inputstring, char * reveredstring)
{
        int length = strlen (inputstring);
        char stack [MAX_LENGTH];
        int top = -1;
```

```
// Push each character on to the stack
for (int i=0 ; i < length ; i++)
{
    stack [++top] = inputString[i] ;
}

// Pop each character from the stack to create the
   reversed string.

int index = 0 ;
while (top != -1 )
{
    reversed String[index ++] = stack [top--];
}
reversedString [index] = '\0' ;
}

int main ()
{
    char inputString[] = " Hello , world!" ;
    char reversedString[MAX_LENGth];

// Reverse the string
reverseString (inputString , reversed string);

// print the reversed string
print ("%s \n", reversedString;
// output : " ! dlro w , olleH "

return 0 ;

}.
```

(3) Write Basic operations of queue.

1) Enqueue (Insertion)
2) Dequeue (Deletion)
3) Front (Peek)
4) IsEmpty
5) IsFull

(4) What is balanced paranthesis.

Balanced parantheses refer to a situation where each opening paranthesis has a corresponding closing parenthesis in the correct order. In other words, for a string containing parentheses, the number of opening parantheses, and they should be properly nested.

(5) Make stack implementation using a linked list - C program.

```
#include <stdio.h>
#include <stlib.h>

// Structure for a stack Note.
struct Node
{
        int data;
        Struct Node* next;
}
// Function to create a new node.
struct Node* createNode (int data)
{
      struct Node* newNode = (struct Node*) malloc (sizeof(
                                                   Struct Node))
   if (newNode == NULL)
{
```

```
Printf ("Memory allocation failed! \n");
exit (1);
}

newNode -> data = data;
newNode -> next = NULL;
return newNode;
}

// structure for the stack
Struct stack
{
        Struct Node * top;
}

// Function to ~~release~~ initialize the Stack
Void initializeStack (struct stack* stack)
{
        stack -> top = NULL;
}

// Function to check if the stack is empty
int isEmpty (struct stack * stack)
{
    return stack -> top == NULL;
}

// Function to push an element to the stack
Void push (struct stack * stack, int data)
{
    struct Node * newNode = createNode (data);
    newNode -> Next = stack -> top;
    stack -> top = newNode;
}
```

```
// Function to pop an element from the stack
int pop (struct Stack * stack)
{
    if (isEmpty (stack))
      {
        printf (" stack underflow ! \n");
        exit (1);
      }

  struct Node * temp = stack -> top;
  int data = temp -> data;
  stack -> top = temp -> next;
  free (temp);
  return data;
}
// Function to get the top element of the stack
int peek (struct Stack * stack)
{
    if (isEmpty (stack))
      {
        printf (" stack is empty ! \n");
        exit (1);
      }
    return stack -> top -> data;
}
int main()
{
   struct Stack stack;
   initializeStack (&stack);

  // pushing element onto the stack
        Push (&stack, 10);
        Push (&stack, 20);
        Push (&stack, 30);
// Print the top element
printf ("Top element : %d \n", peek (&stack));

// Popping elements from the stack
Printf(" Popped element : %d \n", pop(&stack));
Printf (" Popped element : %d \n", pop (& stack));

// Check if the stack is empty
Printf (" Is the stack empty? %s \n",
            isEmpty (&stack) ? "Yess" : "No");

return 0;
}
```

2023/06/21

Tutorials ⑧

① What is a circular queue?

A circular queue also known as "Ring Buffer" is a data structure that follows first-in-first-out (FIFO) principle.

It is implemented as an array / queue with a fixed, size, where the last position is connected to the first position / first index forming a circular behaviour.

② What are the characteristics of circular queue?

- font and rear as pointers : In circular queues two pointers ~~front~~ maintain the order of the circular queue.

- The circular behaviour. : When the data Structure has reached its capacity the next