

CBSE Class XII Computer Science

Stack Questions & Solutions (Board Examinations Compilation)

Question 1: Assertion & Reasoning - Stack Implementation

Assertion (A): In Python a stack can be implemented using a list.

Reasoning (R): A stack is an ordered linear list of elements that works on the principle of First In First Out (FIFO).

Choose the correct option:

- (a) Both (A) and (R) are true and (R) is the correct explanation for (A).
- (b) Both (A) and (R) are true and (R) is not the correct explanation for (A).
- (c) (A) is true but (R) is false.
- (d) (A) is false but (R) is true.

Answer:

(c) (A) is true but (R) is false.

Question 2: Assertion & Reasoning - LIFO Structure

Assertion (A): A stack is a LIFO structure.

Reason (R): Any new element pushed into the stack always gets positioned at the index after the last existing element in the stack.

Choose the correct option:

- (a) Both (A) and (R) are true and (R) is the correct explanation for (A).
- (b) Both (A) and (R) are true and (R) is not the correct explanation for (A).
- (c) (A) is true but (R) is false.
- (d) (A) is false but (R) is true.

Answer:

(c) (A) is true but (R) is false.

Question 3: Stack Implementation - University Course Records

A list contains following record of course details for a University:

[Course_name, Fees, Duration]

Write the following user defined functions to perform given operations on the stack named 'Univ':

- (i) Push_element() - To push an object containing the Course_name, Fees and Duration of a course, which has fees greater than 100000 to the stack.
- (ii) Pop_element() - To pop the object from the stack and display it. Also, display "Underflow" when there is no element in the stack.

For example:

If the lists of courses details are:

["MCA", 200000, 3]

["MBA", 500000, 2]

["BA", 100000, 3]

The stack should contain:

["MBA", 500000, 2]

["MCA", 200000, 3]

Answer:

```
Univ = []

def Push_element(Course):
    for Rec in Course:
        if Rec[1] > 100000:
            Univ.append(Rec)

def Pop_element():
    while len(Univ) > 0:
        print(Univ.pop())
    else:
        print("Underflow")
```

Question 4: Stack Implementation - Hotel Customer Records

A list contains following record of customer:

[Customer_name, Room_Type]

Write the following user defined functions to perform given operations on the stack named 'Hotel':

(i) Push_Cust() - To Push customers' names of those customers who are staying in 'Delux' Room Type.

(ii) Pop_Cust() - To Pop the names of customers from the stack and display them. Also, display "Underflow" when there are no customers in the stack.

For example, if the lists with customer details are as follows:

["Siddarth", "Delux"]

["Rahul", "Standard"]

["Jerry", "Delux"]

The stack should contain:

Jerry

Siddharth

The output should be:

Jerry

Siddharth
Underflow

Answer:

```
Hotel = []
Customer = [["Siddharth", "Delux"], ["Rahul", "Standard"], ["Jerry", "Delux"]]

def Push_Cust():
    for rec in Customer:
        if rec[1] == "Delux":
            Hotel.append(rec[0])

def Pop_Cust():
    while len(Hotel) > 0:
        print(Hotel.pop())
    else:
        print("Underflow")
```

Question 5: Stack Implementation - Vehicle Dictionary

Write a function in Python, Push(Vehicle) where Vehicle is a dictionary containing details of vehicles (Car_Name: Maker).

The function should push the name of car manufactured by 'TATA' (including all the possible cases like Tata, TaTa, etc.) to the stack.

For example, if the dictionary contains the following data:

Vehicle = {"Santro": "Hyundai", "Nexon": "TATA", "Safari": "Tata"}

The stack should contain:

Safari

Nexon

Answer:

```
stack = []

def Push(Vehicle):
    for v_name in Vehicle:
        if Vehicle[v_name].upper() == "TATA":
            stack.append(v_name)
```

Question 6: Stack Implementation - Filtering Big Numbers

Consider a list named Nums which contains random integers.

Write the following user defined functions in Python and perform the specified operations on a stack named BigNums:

- (i) PushBig(): It checks every number from the list Nums and pushes all such numbers which have 5 or more digits into the stack, BigNums.
- (ii) PopBig(): It pops the numbers from the stack, BigNums and displays them. The function should also display "Stack Empty" when there are no more numbers left in the stack.

For example: If the list Nums contains the following data:

Nums = [213, 10025, 167, 254923, 14, 1297653, 31498, 386, 92765]

Then on execution of PushBig(), the stack BigNums should store:

[10025, 254923, 1297653, 31498, 92765]

And on execution of PopBig(), the following output should be displayed:

92765

31498

1297653

254923

10025

Stack Empty

Answer:

```
def PushBig(Nums, BigNums):
    for N in Nums:
        if len(str(N)) >= 5:
            BigNums.append(N)

def PopBig(BigNums):
    while BigNums:
        print(BigNums.pop())
    else:
        print("Stack Empty")
```

Question 7: Stack Implementation - City Dictionary by State Length

A dictionary, d_city contains the records in the following format:

{state: city}

Define the following functions with the given specifications:

- (i) push_city(d_city): It takes the dictionary as an argument and pushes all the cities in the stack CITY whose states are of more than 4 characters.
- (ii) pop_city(): This function pops the cities and displays "Stack empty" when there are no more cities in the stack.

Answer:

```
CITY = []
```

```
def push_city(d_city):
    for c in d_city:
        if len(c) > 4:
            CITY.append(d_city[c])

def pop_city():
    while CITY:
        print(CITY.pop())
    else:
        print("Stack empty")
```

Question 8: Stack Implementation - Keyboard Records & isEmpty

A stack named KeyStack contains records of some computer keyboards. Each record is represented as a list containing Make, Keys, Connectivity. The Make and Connectivity are strings, and Keys is an integer. For example, a record in the stack may be ('Hitech', 105, 'USB').

Write the following user-defined functions in Python to perform the specified operations on KeyStack:

- (i) push_key(KeyStack, new_key): This function takes the stack KeyStack and a new record new_key as arguments and pushes this new record onto the stack.
- (ii) pop_key(KeyStack): This function pops the topmost record from the stack and returns it. If the stack is already empty, the function should display the message "Underflow".
- (iii) isEmpty(KeyStack): This function checks whether the stack is empty. If the stack is empty, the function should return True, otherwise the function should return False.

Answer:

```
def push_key(KeyStack, new_Key):
    KeyStack.append(new_Key)

def pop_key(KeyStack):
    if not KeyStack:
        print("Underflow")
    else:
        return KeyStack.pop()

def isEmpty(KeyStack):
    return KeyStack == []
```

Question 9: Stack Implementation - Vowels from String

Write the following user-defined functions in Python:

- (i) push_vowels(S, St): Here S is a string and St is a list representing a stack. The function should push all the vowels of the string S onto the stack St. For example, if the string S is "Easy Concepts", then the function push_vowels() should push the elements 'E', 'a', 'o', 'e' onto the stack.
- (ii) pop_one(St): The function should pop an element from the stack St, and return this element. If the stack is empty, then the function should display the message 'Stack Underflow', and return None.

(iii) `display_all(St)`: The function should display all the elements of the stack `St`, without deleting them. If the stack is empty, the function should display the message 'Empty Stack'.

Answer:

```
def push_vowels(S, St):
    vowels = 'AEIOUaeiou'
    for ch in S:
        if ch in vowels:
            St.append(ch)

def pop_one(St):
    if len(St) == 0:
        print("Stack Underflow")
        return None
    else:
        return St.pop()

def display_all(St):
    if len(St) == 0:
        print("Empty Stack")
    else:
        print("Stack contents:", St)
```

Question 10: Stack Implementation - Color Records (Tuples)

A stack, named `ClrStack`, contains records of some colors. Each record is represented as a tuple containing four elements `ColorName`, `RED`, `GREEN`, `BLUE`. `ColorName` is a string, and `RED`, `GREEN`, `BLUE` are integers. For example, a record in the stack may be ('Yellow', 237, 250, 68).

Write the following user-defined functions in Python to perform the specified operations on `ClrStack`:

- (i) `push_Clr(ClrStack, new_Clr)`: This function takes the stack `ClrStack` and a new record `new_Clr` as arguments and pushes this new record onto the stack.
- (ii) `pop_Clr(ClrStack)`: This function pops the topmost record from the stack and returns it. If the stack is already empty, the function should display the message "Underflow".
- (iii) `isEmpty(ClrStack)`: This function checks whether the stack is empty. If the stack is empty, the function should return `True`, otherwise the function should return `False`.

Answer:

```
def push_Clr(ClrStack, new_Clr):
    ClrStack.append(new_Clr)

def pop_Clr(ClrStack):
    if len(ClrStack) == 0:
        print("Underflow")
    else:
        return ClrStack.pop()

def isEmpty(ClrStack):
```

```
if len(ClrStack) == 0:
    return True
else:
    return False
```

Question 11: Stack Implementation - Push Trail Elements (Last 5 Items)

Write the following user-defined functions in Python:

- (i) `push_trail(N, myStack)`: Here `N` and `myStack` are lists, and `myStack` represents a stack. The function should push the last 5 elements from the list `N` onto the stack `myStack`. For example, if the list `N` is `[1, 2, 3, 4, 5, 6, 7]`, then the function `push_trail()` should push the elements 3, 4, 5, 6, 7 onto the stack. Therefore the value of stack will be `[3, 4, 5, 6, 7]`. Assume that `N` contains at least 5 elements.
- (ii) `pop_one(myStack)`: The function should pop an element from the stack `myStack`, and return this element. If the stack is empty, then the function should display the message 'Stack Underflow', and return `None`.
- (iii) `display_all(myStack)`: The function should display all the elements of the stack `myStack`, without deleting them. If the stack is empty, the function should display the message 'Empty Stack'.

Answer:

```
def push_trail(N, myStack):
    for i in range(-5, 0, 1):
        myStack.append(N[i])

def pop_one(myStack):
    if not myStack:
        print('Stack Underflow')
    else:
        return myStack.pop()

def display_all(myStack):
    if not myStack:
        print("Empty Stack")
    else:
        for i in myStack[::-1]:
            print(i, end=' ')
```

Question 12: Stack Implementation - Fruit Records with Price Condition

A stack named `FruitStack`, implemented using list, contains records of some fruits. Each record is represented as a dictionary with keys 'Name', 'Origin', 'Price' and 'Expiry'. A sample record is given here:

```
{'Name': 'Apple', 'Origin': 'France', 'Price': 120, 'Expiry': '12-08-2025'}
```

Write the following user-defined functions in Python to perform the specified operations on `FruitStack`:

- (i) `push_fruit(FruitStack, Fruit)`: This function takes the stack `FruitStack` and a new record `Fruit` as

arguments and pushes the record stored in Fruit onto FruitStack if the Price is less than 100.

(ii) pop_fruit(FruitStack): This function pops the topmost record from the stack and returns it. If the stack is already empty, the function should display "UNDERFLOW".

(iii) display(FruitStack): This function displays all the elements of the stack starting from the topmost element. If the stack is empty, the function should display 'EMPTY STACK'.

Answer:

```
def push_fruit(FruitStack, Fruit):
    if Fruit['Price'] < 100:
        FruitStack.append(Fruit)

def pop_fruit(FruitStack):
    if FruitStack == []:
        print('UNDERFLOW')
    else:
        return FruitStack.pop()

def display(FruitStack):
    if FruitStack == []:
        print('EMPTY STACK')
    else:
        for ele in FruitStack[::-1]:
            print(ele)
```

Question 13: Stack Program - Three-digit Even Integers

Write a Python program to accept 10 integers from the user. If the entered number is a three digit even integer, push it onto a stack. After all inputs are taken, pop all the three-digit even integers from the stack and display them.

For example, if the user enters:

12, 31, 320, 457, 6, 92, 924, 220, 1, 218

Then the stack should contain:

320, 924, 220, 218

And the output of the program should be:

218 220 924 320

Answer:

```
Stack = []

for C in range(10):
    Num = int(input("Integer: "))
    if Num >= 100 and Num <= 999 and Num % 2 == 0:
        Stack.append(Num)
```

```
while Stack:
    print(Stack.pop(), end=' ')
```

Question 14: Stack Implementation - High Performers

Consider a globally declared dictionary named `EMPLOYEE_PERFORMANCE` which contains employee IDs and their respective performance scores (out of 100) as key:value pairs:

```
EMPLOYEE_PERFORMANCE = {
    "E101": 85,
    "E102": 92,
    "E103": 78,
    "E104": 95,
    "E105": 80,
    "E106": 70
}
```

Write the following user-defined functions to perform the operations on a globally declared list implementing a stack named `TOP_PERFORMERS_LIST`:

- (I) `PushHighPerformers()`: This function should push all employee IDs from the dictionary `EMPLOYEE_PERFORMANCE` whose performance score is greater than or equal to 90 into the stack `TOP_PERFORMERS_LIST`.
- (II) `IsPerformanceListEmpty()`: This function should check whether the stack `TOP_PERFORMERS_LIST` is empty or not. If it is empty, the function should return `True`; otherwise, it should return `False`.
- (III) `PopPerformer()`: This function should pop and return the topmost employee ID from the stack `TOP_PERFORMERS_LIST`. If the stack is already empty (check using `IsPerformanceListEmpty()`), then display "Stack Underflow: No top performer to pop."

Answer:

```
def PushHighPerformers():
    for K, V in EMPLOYEE_PERFORMANCE.items():
        if V >= 90:
            TOP_PERFORMERS_LIST.append(K)

def IsPerformanceListEmpty():
    if len(TOP_PERFORMERS_LIST) == 0:
        return True
    else:
        return False

def PopPerformer():
    if IsPerformanceListEmpty():
        print("Stack Underflow: No top performer to pop.")
        return None
    else:
        return TOP_PERFORMERS_LIST.pop()
```

Question 15: Stack Implementation - Discount Prices (LIFO Order)

Consider a globally declared list of product prices named `PRODUCT_PRICES` as follows:

```
PRODUCT_PRICES = [150.75, 85.00, 299.99, 50.50, 120.00, 99.95, 450.00]
```

Write the following user-defined functions to perform operations on a globally declared list implementing a stack named `DISCOUNT_PRICES`:

(I) `PushDiscPrices()`: Pushes all prices from `PRODUCT_PRICES` which are less than or equal to 100.00 into `DISCOUNT_PRICES`.

(II) `PopDiscPrices()`: Pops and displays the last (topmost) price from `DISCOUNT_PRICES`. If the stack is empty, displays "No More Discount Prices to Pop" and returns `None`.

(III) `ShowDiscPrices()`: Displays all prices from `DISCOUNT_PRICES` without deleting them in LIFO order (last to first). If empty, displays "Stack is Empty: No discount prices to show".

Answer:

```
def PushDiscPrices():
    for price in PRODUCT_PRICES:
        if price <= 100.00:
            DISCOUNT_PRICES.append(price)

def PopDiscPrices():
    if DISCOUNT_PRICES == []:
        print("No More Discount Prices to Pop")
        return None
    else:
        PoppedPrice = DISCOUNT_PRICES.pop()
        print("Popped", PoppedPrice)
        return PoppedPrice

def ShowDiscPrices():
    if not DISCOUNT_PRICES:
        print("stack is Empty: No discount prices to show.")
    else:
        print("Contents of DISCOUNT_PRICES (LIFO Order):")
        for V in DISCOUNT_PRICES[::-1]:
            print(V)
```