1.1.2024

1. WAP to simulate the working of stack using an array to show
(i) push
(ii) pull
(iii) display
the program should print appropriate messages for stack overflow, stack underflow.

CODE:-

```
# include <stdio.h>
# include <stdlib.h>
# define size 5
void push(int);
void pop();
void display();
int stack[size], top=-1;
void main(){
  int op, n;
  printf("Enter the operation\n 1. push\n 2. pop\n 3. display\n
      Enter -1 to stop\n");
  while(1) {
    scanf("%d", &op);
    if(op==-1){
    printf("Stopping the operations\n");
    break;
    }
    else {
     switch(op){
     case 1:
      printf("Enter the value\n");
      scanf("%d", &n);
      push(n);
      break;

     case 2:
```

```
        pop();
        break;
      case 3:
        display();
        break;
      default:
        printf("Wrong choice\n");
      }
    }
  }
}

void push(int x) {
    if (top == size-1) {
      printf("Stack overflow condition\n");
    }
    else {
      top++;
      stack[top] = x;
      printf("PUSH() Operation is successful\n");
    }
  }

void pop() {
  if (top == -1) {
      printf("Stack underflow condition\n");
  }
  else {
      printf("%d is
      printf("Stack pop() operation successfully\n", stack[top]);
      top--;
    }
  }
void display () {
  if (top == -1) {
      printf("Stack is empty");
```

Q2.

```
Enter the operation:
1.push
2.pop
3.display
4.-1 to stop
1
Enter the number:
5
successfully pushed
Enter the operation:
1.push
2.pop
3.display
4.-1 to stop
1
Enter the number:
6
successfully pushed
Enter the operation:
1.push
2.pop
3.display
4.-1 to stop
1
Enter the number:
7
successfully pushed
Enter the operation:
1.push
2.pop
3.display
4.-1 to stop
1
Enter the number:
8
stack overflow
Enter the operation:
1.push
2.pop
3.display
4.-1 to stop
```

```
}
else {
  for (int i = top; i >= 0; i--) {
      printf("%d \t", stack[i]);
  }
  printf("\n");
  }
}
```

OUTPUT :-

Enter the operation 1. push
2. pop
3. display
4. -1 to exit

1

Enter the number

7

Successfully pushed

Q2. WAP to convert infix to postfix expression.

```
#include <stdio.h>
#include <string.h>
int index1=0, pos=0, top=-1, length;
char symbol, temp, infix[20], postfix[20], stack[20];
void infixtopostfix();
void push(char symbol);
char pop();
int pred(char symbol);
void main(){
    printf("Enter the infix expression \n");
    scanf("%s", infix);
     infixtopostfix();
    printf("Infix expression: %s\n", infix);
    printf("Postfix expression: %s\n", postfix);
    }
void infixtopostfix(){
    length = strlen(infix);
    push('#');
    while(index1 < length){
        symbol = infix[index1];
        switch(symbol){
          case '(': push(symbol);
                    break;
          case ')': temp = pop();
```

8.1.2024

3.

```
while(temp != '(') {
    postfix[pos] = temp;
    pos++;
    temp = pop();
}
break;
case '+': push()
case '-': push()
case '*': push()
case '/': push();
case '^': push()
while ( pred(stack[top]) >= pred(symbol)) {
    temp = pop();
    postfix[pos++] = temp;
}
}
void push(char symbol){
```

OUTPUT

(K+L-M*N+(O^P)*W/U/V*T+Q)

Infix expression : (K+L-M*N+(O^P)*W/U/V*T+Q)

Postfix expression : KL+MN*-OP^W*U/V/T*+Q+

Ended 1/1/2024

```
Enter the infix expression
(k+l-m*n+(o^p)*w/u/v*t+q)
Infix expression:(k+l-m*n+(o^p)*w/u/v*t+q)
Postfix expression:kl+mn*-op^w*u/v/t*+q+

Process returned 41 (0x29)   execution time : 39.049 s
Press any key to continue.
```