01-01-2024

(a) Write a program to stimulate the working of stack using an array with the following:
(a) Push (b) Pop (c) Display.
The program should print appropriate messages for stack overflow, stack underflow

I/p:-

```
void push(int x)
{
 if(top==x-1
int top=-1;
 int stack[size];

void push(int x)
{
  if (top==size-1)
    {
    printf("stack overflow: cannot
        push element. stack is
            full");
    }
 else
    {
     stack[++top]=x;
     printf("value %d is pushed to
                stack", x);
    }
}
```

```
void pop(int x)
{
  if (top==-1){
    printf("Stack underflow cannot pop
            element, Stack is empty");
  }
  else {
    printf("The element %d is deleted
            stack[top]);
    top= top-1;
  }
}
void display(int x)
{
  if (top==-1)
  {
    printf("The Stack is empty");
  }
  else
  {
    printf("The Stack elements are:");
    for(int i=0; i<=top; ++i)
    {
      printf("%d", stack[i]);
    }
    print("\n");
  }
}
```

6. WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression the expression consist of single character operands and binary operators +, -, *, /.

```
I/p:- int index=0, pos=0, top=-1;
      char infix[20], postfix[20], stack[20];

      void infixtopostfix()
       {
        length=strlen(infix);
        push('H');
        while (index<length)
         {
          symbol=infix[index];
          switch(symbol)
          {
          case '(': push(symbol);
                    break;
          case ')': temp=pop();
                    while(temp!='(')
                    {
                     postfix[pos]=temp;
                     pos++;
                     temp=pop();
                    }
                    break;
          case '+':
          case '-':
          case '*':
          case '(':
```

```
        while (pred (stack[top]) >= pred(symbol))
        {
         temp= pop();
         postfix [post++]= temp;
          }
          push(symbol);
          break;
    default: postfix[post++]= symbol;
     }
      index++;
  }
   while (top>0)
    {
      temp= pop();
      postfix [post++]= temp;
    }
  }
void push (char symbol)
 {
  top= top+1;
  stack[top] = symbol;
 }
  char pop()
  {
  char symb;
  symb = stack[top];
   top = top-1;
   return (symb);
  }.
```

```
int pred (char symbol)
{
  int p;
  switch (symbol)
  {
    case '^': p=3;
       break;
    case '*':
    case '/': p=2;
       break;
    case '+':
    case '-': p=1;
       break;
    case '(': p=0;
       break;
    case '#': p=-1;
       break;
  }
  return (p);
}
```

01/01/24