Q. Implement push, pop and find the minimum element in a stack in O(1) time complexity

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

int stack[100], support_stack[100];

int push(int element, int *top, int *stack)
{
    *top = *top + 1;
    stack[*top] = element;
}

int pop(int *stack, int *top)
{
    int element;
    if (*top > -1)
    {
        element = stack[*top];
        *top = *top-1;
        return element;
    }
```

```
    else
    {
        printf ("\n STACK EMPTY\n");
        returns -99999; // means nothing is popped.
    }
}
int main()
{
    int choice, element, top_main = -1;
    int top_support = -1, i, supp_pop_element;
    int pop_element;
    printf (" Enter the operation:\n");
    printf ("\n1. Push\n 2. Pop\n 3. Check min\n");
    printf ("4. Stop\n");
    scanf ("%d", &choice);
    while (choice != 5)
    {
        if (choice == 1)
        {
            printf ("\nEnter num");
            scanf ("%d", &element);
```



```
            push(element, &top_main, main_stack);
            if(top_support >= 0 && element < support_stack[top_supp
            {
                push(element, &top_support, support_stack);
            }
            else if(top_support == -1)
            {
                push(element, &top_support, support_stack);
            }
        }
        else if (choice == 2)
        {
            pop_element = pop(stack, &top_main);
            if(pop_element != -99999)
                printf("\n Popped elent = %d", pop_elent);
            if(pop_element != -99999)
            {
                if(pop_element == support_stack[top_suppo
                {
                    supp_pop_elent = pop(support_stack, &top_s-pp
                }
            }
        }
    }
```

```
else if (choice == 3)
{
  if(top_support > -1)
   printf("\n Min elent = %d\n", support_stack[top_sup]);
  else
   printf("\n STACK EMPTY");
}

else if (choice == 4)
{
  if (top_main > -1)
  {
     printf ("\n MAIN STACK\n");
     for(i = top_main ; i >= 0; i--)
     {
       printf ("\n%d", stack [i]);
     }
  }
else
   printf ("\n STACK EMPTY\n");
   printf ("\n top_support = %d", top_support);
   for (i = top_support ; i >= 0; i--)
      {
        printf ("\n%d", support_stack[i]);
      }
}
```

```
printf("\n Enter the operation : \n 1. Push\n");
printf("\n 2. Pop \n 3. check minimum\n 4. See full stack\n");
printf("\n 5. STOP \n");

scanf(" %d", &choice);
}
return 0;
}
```