URBAN EDGE 21/12/23

LAB-02

1) Swapping using pointers:

```
#include <stdio.h>
void swapref (int * a , int *b)
void main()
{
    int a=10;
    int b= 15;
    printf ("the values of a and b before swapping are: \n%.d
            \n %.d", a,b);
    swapref (&a, &b)
    printf (" swapping of two values after call by reference \n%.d
            \n %.d", a,b);
}
Void swapref (int * a, int* b)
{
    int temp;
    temp = * a;
    *a = *b;
    * b = temp;
    printf ("the values of a and b after swapping: \n%.d\n%.d",
            a,b);
}
```

Output:-

The values of a and b before swapping:

10

15

The values of a and b after swapping:

15

10

2) Dynamic memory:

```
#include <stdio.h>
Void MallocEx (int);
Void callocEx (int);
Void main()
{
    int n;
    printf("Enter the value of n : \n");
    scanf("%d", &n);
    MallocEx (n);
    callocEx (n);
}
Void MallocEx (int n)
{
    int *ptr;
    int i;
    int arr[n];
    ptr = (int *) malloc (n * sizeof(int));
    for (i=0; i<n; i++)
        ptr[i] = i+1;
    printf("malloc dynamic memory allocation \n");
    printf("the elements of the array are : \n");
    for (i=0; i<n; i++)
        printf("%d", ptr[i]);
    printf("\n");
    free (ptr)
}
Void callocEx (int n)
{
    int *ptr;
    int i;
    int arr[n];
    ptr = (int *) calloc (n, sizeof(int));
    for (i=0; i<n; i++)
        printf("%d", ptr[i]);
    printf("calloc dynamic memory allocation \n");
    printf("the elements of the array are : \n");
    n=15
    ptr = (int *) realloc (ptr, n * sizeof(int));
    for (i=10; i<n; i++)
        ptr[i] = i+1;
    for (i=0; i<n; i++)
        printf("%d", ptr[i]);
    free (ptr);
}
```

Output:

```
Enter the value of n:
7
Malloc dynamic memory allocation
the elements of the array are:
1 2 3 4 5 6 7
Calloc dynamic memory allocation
the elements of the array are:
1 2 3 4 5 6 7
Realloc dynamic memory allocation
the elements of the array are
1 2 3 4 5 6 7 0 58 7 2025 9325279 11 2131415
```

3) Stack implementation:

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
int top = -1;
int inp_array[SIZE];
Void push()
{
    int x;
    if (top == SIZE -1)
        printf("overflow\n");
    else
    {
        printf("Enter the element to be added
                in the stack : \n");
        scanf("%d", &x);
        top = top + 1;
        inp_array[top] = x;
    }
}
void pop()
{
    if (top == -1)
        printf("underflow \n");
    else
    {
        printf("Enter the element to be removed : \n");
        scanf("%d", &x);
        printf("popped element is %d \n", inp_array[top]);
        top = top - 1;
    }
}
```

```
Void show()
{
    if (top == -1)
        printf("underflow \n");
    else
    {
        printf("Elements in the stack are : \n");
        for (int i = top; i >= 0; --i);
            printf("%d \n", inp_array[i]);
    }
}
```

output:

```
operation on the stack
1. push the element
2. pop the element
3. show

Enter the choice:
3
underflow
operation on the stack:
1. push the element
2. pop the element
3. show
4. End
Enter the choice
1
Enter the element to be added in the stack
5
```

Operations on the stack

1. push the element
2. pop the element
3. show
4. End

Enter the choice:

1

Enter the element to be added in stack:

3

Operations on the stack:

1. push the element
2. pop the element
3. show
4. End

Enter the choice:

2.

Enter the elements to be removed:

3

~~P~~opped element : 3