Assignment-2

1. Write a menu driven program to create/manage the action list for the following problem: the Processor is keeping record of all the actions taken by the user. If the user press “UnDo” the control should go to the last action to discard it, and if the user “REDO” the last action is to be redone. (If no records in processor then check underflow condition and if records are beyond size limit then check overflow condition)

Code -

```c
#include <stdio.h>

void redo(int arr[], int * top, int *val){
    if(*top == 6){
        printf("Already Filled !");
    }
    else if(*val == -2){
        printf("Redo only once!");
    }
    else{
        *top = *top + 1;
        arr[*top] = *val;
        *val = -2;
    }
}

int undo(int arr[], int * top){
    if(*top == -1){
        return 0;
    }
    else{
        int val = arr[*top];
        *top = *top - 1;
        return val;
    }

}
```

```c
void add(int arr[], int * top){
    int m;
    if((*top)>=6){
        printf("Stack Overflow Error !");
    }
    else{
        printf("Enter the no. of elements to add - ");
        scanf("%d", &m);
        if(m > (6-(*top))){
            printf("Stack Overflow Error can occur ! Only %d
elements can be added at max", 6-(*top));
        }
        else{

            while(m--){
              *top += 1;
              scanf("%d", &arr[*top]);

            }
        }
    }
}

int main(){
    int arr[7], top=-1, action = 0, val;
    printf("Enter the 7 elements array to perform action on -
");
    for(int i =0;i<7;i++){
        scanf("%d", &arr[i]);
        top+=1;
    }

    while(top<=7){
        printf("\nThese are the Actions you can perform on the
array : [ ");
        if(top == -1){
            printf("Empty");
        }
        else{
```

```
            for(int i=0;i<=top;i++){
                printf("%d ", arr[i]);
            }
        }

        printf("]\n");
        printf("1. Add\n2. Undo\n3. Redo\n0. Exit\n --> ");
        scanf("%d", &action);
        switch(action){
            case 1:
                add(arr, &top);
                break;
            case 2:
                val = undo(arr, &top);
                break;
            case 3:
                redo(arr, &top, &val);
                break;
            case 0:
                top = 9;
                break;
            default:
                break;
        }
    }
}
```

Output -

```
Enter the 7 elements array to perform action on - 1 2 3 4 5 6 7

These are the Actions you can perform on the array : [ 1 2 3 4 5 6 7 ]
1. Add
2. Undo
3. Redo
0. Exit
 --> 2

These are the Actions you can perform on the array : [ 1 2 3 4 5 6 ]
1. Add
2. Undo
3. Redo
0. Exit
 --> 2

These are the Actions you can perform on the array : [ 1 2 3 4 5 ]
1. Add
2. Undo
3. Redo
0. Exit
 --> 2

These are the Actions you can perform on the array : [ 1 2 3 4 ]
1. Add
2. Undo
3. Redo
0. Exit
 --> 3

These are the Actions you can perform on the array : [ 1 2 3 4 5 ]
1. Add
2. Undo
3. Redo
0. Exit
 --> 1
Enter the no. of elements to add - 9
Stack Overflow Error can occur ! Only 2 elements can be added at max
These are the Actions you can perform on the array : [ 1 2 3 4 5 ]
1. Add
2. Undo
3. Redo
0. Exit
 --> 0
```

2. Given an expression string exp, write a program to examine whether the pairs and the orders of “{“, “}”, “(“, “)”, “[“, “]” are correct in the given expression.

   Example: **Input**: exp = “[()]{}{[()()]()}”

   Output: Balanced

   Explanation: *all the brackets are well-formed*

   Input: exp = “[(])”

   Output: Not Balanced

   Code -

```c
#include <stdio.h>
#include<string.h>
#include <stdlib.h>

#define MAX 50

struct stack {
  char stck[MAX];
  int top;
}s;

void push(char value) {
  if (s.top == (MAX - 1))
    printf("Stack is Full--OVERFLOW\n");

  else {
    s.top = s.top + 1;
    s.stck[s.top] = value;
  }
}

void pop() {
  if (s.top<0)
    printf("Stack is Empty--UNDERFLOW\n");

  else
```

```
    s.top = s.top - 1;
}

int check_pair(char exp1,char exp2){
    return (( exp1=='(' && exp2==')' )||( exp1=='[' &&
exp2==']' )||( exp1=='{' && exp2=='}' ));
}

int check_balancing(char exp[], int len){

    for (int i = 0; i < len; i++)
    {
        if (exp[i] == '(' || exp[i] == '[' || exp[i] == '{')
        {
          push(exp[i]);
        }
        else
        {

            if (s.top == -1)
                return 0;
            else if(check_pair(s.stck[s.top],exp[i]))
            {
                pop();
                continue;
            }

            return 0;
        }
    }
    return 1;
}


int main() {
  char brc[MAX];
  printf("Enter the expression: ");
  gets(brc);
  int i = 0;
```

```
  s.top = -1;

  int len = strlen(brc);
  int x=check_balancing(brc, len);
  if(x==1){
    printf("The exp is balanced--\n");
  }else{
    printf("The exp is not balanced--\n");
  }

  return 0;
}
```

Output -

```
warning: this program uses gets(), which is unsafe.
Enter the expression: {[({{}})]}
The exp is balanced--
garvitshah@Garvits-MacBook-Air ~/D/C/D/Lab> cd "/Users/garvi
tshah/Desktop/College/DS/Lab/"Assign2
warning: this program uses gets(), which is unsafe.
Enter the expression: {{[()]}}
The exp is not balanced--
```