/\* Given an array of nonnegative integers, where all numbers occur even number of times except one number which occurs odd number of times. Write an algorithm and a program to find this number. (Time complexity = O(n))

Name – Ayush Barthwal

Roll no. - 18

Section - C1

Course – B.TECH

Branch – CSE Sem III

\*/

#include <stdio.h>

int Odd(int arr[], int n) {

int result = 0;

for (int i = 0; i < n; i++) {

result ^= arr[i];

}

return result;

}

int main(){

int t;

scanf("%d",&t);

for (int i=0; i<t; i++){

int n;

scanf("%d",&n);

int arr[n];

for (int j = 0; j<n; j++){

scanf("%d",&arr[j]);

}

printf("Number - %d\n", Odd(arr, n));

}

}

OUTPUT

Testcase1-

1

5

2 4 3 2 3

Number - 4

Testcase2-

2

11

1 6 3 2 4 1 4 2 3 6 6

Number – 6

15

2 8 7 1 2 6 8 9 0 2 6 2 9 7 1

Number - 0

/\* Design an algorithm and a program to implement stack using array. The program should implement following stack operations:

a) Create() - create an empty stack

b) Push(k) - push an element k into the stack

c) Pop() - pop an element from the stack snd return it

d) IsEmpty() - check if stack is empty or not

e) Size() - finds the size of the stack

f) Print() - prints the contents of stack

Name – Ayush Barthwal

Roll no. - 18

Section - C1

Course – B.TECH

Branch – CSE Sem III

\*/

#include <stdio.h>

int top = -1;

void Create(int stack[], int n) {

top = -1;

}

void Push(int stack[], int k, int n) {

if (top == n - 1) {

printf("ERROR! Stack Overflow.\n");

} else {

top++;

stack[top] = k;

printf("Added element %d to the stack.\n", k);

}

}

int Pop(int stack[]) {

if (top == -1) {

printf("ERROR! Stack Underflow.\n");

return -1;

} else {

int poppedElement = stack[top];

top--;

printf("Successfully popped element %d from the stack.\n", poppedElement);

return poppedElement;

}

}

int IsEmpty() {

return top == -1;

}

int Size() {

return top + 1;

}

void Print(int stack[]) {

if (IsEmpty()) {

printf("Stack is empty.\n");

} else {

printf("Stack contents: ");

for (int i = 0; i <= top; i++) {

printf("%d ", stack[i]);

}

printf("\n");

}

}

int main() {

int n;

printf("Enter the maximum size of the stack: ");

scanf("%d", &n);

int stack[n];

int end = 1;

while (end != 0) {

int choice;

printf("\n1. Push element\n2. Pop element\n3. Show stack\n4. Check if empty\n5. Get size\n6. End");

printf("\n\nEnter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: {

int k;

printf("Enter the value you want to push: ");

scanf("%d", &k);

Push(stack, k, n);

break;

}

case 2:

Pop(stack);

break;

case 3:

Print(stack);

break;

case 4:

if (IsEmpty()) {

printf("Stack is empty.\n");

} else {

printf("Stack is not empty.\n");

}

break;

case 5:

printf("Current size of the stack: %d\n", Size());

break;

case 6:

end = 0;

break;

default:

printf("Invalid choice!!\n");

}

}

return 0;

}

OUTPUT

Testcase1-

1

5

2 4 3 2 3

Number - 4

Testcase2-

2

11

1 6 3 2 4 1 4 2 3 6 6

Number – 6

15

2 8 7 1 2 6 8 9 0 2 6 2 9 7 1

Number - 0

/\* Given an expression string consisting of opening and closing brackets “{“,”}”,”(“,”)”,”[“,”]”, design an algorithm and a program to check whether this expression has balanced paranthesis or not.

Name – Ayush Barthwal

Roll no. - 18

Section - C1

Course – B.TECH

Branch – CSE Sem III

\*/

#include <stdio.h>

#include <string.h>

int *check*(char *s*[], int *l*){

    int count1 = 0, count2 = 0, count3 = 0;

    for (int j = 0; j < l; j++){

        if (s[j] == '(') count1++;

        if (s[j] == ')') count1--;

        if (s[j] == '[') count2++;

        if (s[j] == ']') count2--;

        if (s[j] == '{') count3++;

        if (s[j] == '}') count3--;

        if (count1 < 0 || count2 < 0 || count3 < 0) return 0;

    }

    if (count1 == 0 && count2 == 0 && count3 == 0) return 1;

    else return 0;

}

int *main*(){

    char s[100];

*fgets*(s,100,stdin);

    int l = *strlen*(s);

    l--;

    if (*check*(s, l)){

*printf*("Balanced");

    }

    else{

*printf*("Unbalanced");

    }

    return 0;

}

OUTPUT

Testcase1-

1

5

2 4 3 2 3

Number - 4

Testcase2-

2

11

1 6 3 2 4 1 4 2 3 6 6

Number – 6

15

2 8 7 1 2 6 8 9 0 2 6 2 9 7 1

Number - 0

/\* Given an empty stack, design an algorithm and a program to reverse a string using this stack (with and without recursion).

Name – Ayush Barthwal

Roll no. - 18

Section - C1

Course – B.TECH

Branch – CSE Sem III

\*/

#include <stdio.h>

#include <string.h>

int top = -1;

void push(char a, char stack[]) {

stack[++top] = a;

}

char pop(char stack[]) {

return stack[top--];

}

void reverseString(char s[]) {

int l = strlen(s);

char stack[l];

for (int i = 0; i < l; i++) {

push(s[i], stack);

}

for (int i = 0; i < l; i++) {

s[i] = pop(stack);

}

}

int main() {

char s[100];

printf("Enter the string: ");

fgets(s, 100, stdin);

s[strcspn(s, "\n")] = '\0';

reverseString(s);

printf("Reversed string: %s\n", s);

return 0;

}

OUTPUT

Testcase1-

1

5

2 4 3 2 3

Number - 4

Testcase2-

2

11

1 6 3 2 4 1 4 2 3 6 6

Number – 6

15

2 8 7 1 2 6 8 9 0 2 6 2 9 7 1

Number - 0