REGISTRATION NUMBER – **22MCA0223**

NAME - **KAMRAN ANSARI**

Q1 - Develop a program to find the parity is EVEN PARITY or ODD PARITY of the given ERROR DETECTION CODE given as the input string. The given input string contains only the binary digits (bits) either 0 or 1. Assume the input string is of the half word length i.e. 16 bits and the last bit is allocated to assign the parity. The parity bit ZERO i.e 0 represents EVEN PARITY and the parity bit ONE i.e 1 represents ODD PARITY.

Program

```
// q1.cpp

#include <iostream>
#include <string.h>
#include <limits.h>

using namespace std;

class Stack
{
private:
    int array[100], n, top;

public:
    Stack()
    {
        top = -1;
        n = 100;
    }
```

```
int size()
{
    return top + 1;
}

bool isFull()
{
    return top >= n - 1;
}

bool isEmpty()
{
    return top <= -1;
}

int push(int value)
{
    if (isFull())
    {
        return INT_MIN;
    }

    array[++top] = value;

    return value;
}

int pop()
```

```
    {
        if (isEmpty())
        {
            return INT_MIN;
        }

        return array[top--];
    }

    int peek()
    {
        if (isEmpty())
        {
            return INT_MIN;
        }

        return array[top];
    }
};

int main()
{
    Stack parityStack;

    string code;
    cout << "Enter the binary code : ";
    cin >> code;

    for (int i = 0; i < code.length(); i++)
```

```
    {
        if (code[i] == '1')
        {
            parityStack.push(code[i]);

            if (parityStack.size() == 2)
            {
                parityStack.pop();
                parityStack.pop();
            }
        }
    }

    if (parityStack.isEmpty())
    {
        cout << "Even Parity";
    }
    else
    {
        cout << "Odd Parity";
    }

    return 0;
}
```

Output Screenshots

```
$ g++ q1.cpp; ./a.exe
Enter the binary code : 1011010001010010
Odd Parity
```

```
$ g++ q1.cpp; ./a.exe
Enter the binary code : 1000100011110000
Even Parity
```

Q3 - Implement a program to perform the Polynomial Addition usingSingly Liked List.

Program

```java
// Term.java

package Lab_Assessment_1.Q3;

public class Term {
    public int coeff, power;
    public Term next;

    Term(int coeff, int power) {
        this.coeff = coeff;
        this.power = power;
        this.next = null;
    }
}
```

```java
// Polynomial.java

package Lab_Assessment_1.Q3;

public class Polynomial {
    private Term head, itr;

    public Polynomial() {
        head = null;
```

```
        itr = null;
    }

    public boolean isEmpty() {
        return head == null;
    }

    public void insertTerm(Term term) {
        this.insertTerm(term.coeff, term.power);
    }

    public void insertTerm(int coeff, int power) {
        Term newTerm = new Term(coeff, power);

        if (head == null) {
            head = newTerm;
            itr = head;
        } else {
            Term itr = head;
            while (itr.next != null) {
                itr = itr.next;
            }

            itr.next = newTerm;
        }
    }

    public void print() {
        Term itr = head;
```

```
        while (itr != null) {
            System.out.print(itr.coeff + "x^" + itr.power
+ " ");
            itr = itr.next;
        }
        System.out.println("");
    }

    public Term currentTerm() {
        return itr;
    }

    public Term nextTerm() {
        if (head == null) {
            return null;
        }

        Term toReturn = itr;
        itr = itr.next;
        return toReturn;
    }

    public boolean hasNextTerm() {
        if (head == null) {
            return false;
        }

        if (itr == null) {
            return false;
```

```
        }

        return true;
    }
}
```

```
// Main.java

package Lab_Assessment_1.Q3;

import java.util.Scanner;

class Main {
    public static void main(String[] args) {
        Polynomial p1 = new Polynomial();
        Polynomial p2 = new Polynomial();

        System.out.println("Enter degree of first
polynomial : ");
        inputPolynomial(p1);

        System.out.println("Enter degree of second
polynomial : ");
        inputPolynomial(p2);

        Polynomial summedPolynomial = addPolynomial(p1,
p2);
```

```
        System.out.println("First Polynomial: ");
        p1.print();

        System.out.println("Second Polynomial: ");
        p2.print();

        System.out.println("Summed Polynomial: ");
        summedPolynomial.print();
    }

    public static void inputPolynomial(Polynomial poly) {
        Scanner scan = new Scanner(System.in);
        int deg = scan.nextInt();

        for (int i = deg; i >= 0; i--) {
            System.out.println("Enter coefficient of x^" +
i + " : ");
            int coeff = scan.nextInt();
            poly.insertTerm(coeff, i);
        }
    }

    public static Polynomial addPolynomial(Polynomial p1,
Polynomial p2) {
        Polynomial summedPolynomial = new Polynomial();

        while (p1.hasNextTerm() && p2.hasNextTerm()) {
            if (p1.currentTerm().power >
p2.currentTerm().power) {
```

```
                Term termToInsert = p1.nextTerm();
                summedPolynomial.insertTerm(termToInsert);
            } else if (p1.currentTerm().power <
p2.currentTerm().power) {
                Term termToInsert = p2.nextTerm();
                summedPolynomial.insertTerm(termToInsert);
            } else {
                Term p1Term = p1.nextTerm();
                Term p2Term = p2.nextTerm();
                Term termToInsert = new Term(p1Term.coeff
+ p2Term.coeff, p1Term.power);
                summedPolynomial.insertTerm(termToInsert);
            }
        }

        while (p1.hasNextTerm()) {
            Term termToInsert = p1.nextTerm();
            summedPolynomial.insertTerm(termToInsert);
        }

        while (p2.hasNextTerm()) {
            Term termToInsert = p2.nextTerm();
            summedPolynomial.insertTerm(termToInsert);
        }

        return summedPolynomial;
    }
}
```

Output Screenshots

```
Enter degree of first polynomial :
1
Enter coefficient of x^1 :
2
Enter coefficient of x^0 :
3
Enter degree of second polynomial :
1
Enter coefficient of x^1 :
1
Enter coefficient of x^0 :
1
First Polynomial:
2x^1 3x^0
Second Polynomial:
1x^1 1x^0
Summed Polynomial:
3x^1 4x^0
```

```
Enter degree of first polynomial :
1
Enter coefficient of x^1 :
3
Enter coefficient of x^0 :
2
Enter degree of second polynomial :
2
Enter coefficient of x^2 :
5
Enter coefficient of x^1 :
6
Enter coefficient of x^0 :
7
First Polynomial:
3x^1 2x^0
Second Polynomial:
5x^2 6x^1 7x^0
Summed Polynomial:
5x^2 9x^1 9x^0
```

Q4 - Develop a program to merge two Linked List and remove the duplicated in it. The linked can be chosen as Singly Linked List or Doubly Linked List as per your choice.

Program

```cpp
// q4.cpp

#include <iostream>

class Node
{
public:
    int value;
    Node *next;

    Node(int value)
    {
        this->value = value;
        this->next = nullptr;
    }
};

class SLList
{
public:
    Node *head;

    SLList()
    {
        head = nullptr;
    }
```

```
bool isEmpty()
{
    return head == nullptr;
}

void insert(int value)
{
    Node *newNode = new Node(value);

    if (head == nullptr)
    {
        head = newNode;
    }
    else
    {
        Node *itr = head;

        while (itr->next != nullptr)
        {
            itr = itr->next;
        }

        itr->next = newNode;
    }
}

void print()
{
```

```
        Node *itr = head;

        while (itr != nullptr)
        {
            std::cout << itr->value << " ";
            itr = itr->next;
        }
    }
};

SLList mergeLists(Node *h1, Node *h2)
{
    SLList listToReturn;

    while (h1 != nullptr && h2 != nullptr)
    {
        if (h1->value < h2->value)
        {
            listToReturn.insert(h1->value);
            h1 = h1->next;
        }
        else if (h1->value > h2->value)
        {
            listToReturn.insert(h2->value);
            h2 = h2->next;
        }
        else
        {
            listToReturn.insert(h1->value);
```

```
            h1 = h1->next;
            h2 = h2->next;
        }
    }

    while (h1 != nullptr)
    {
        listToReturn.insert(h1->value);
        h1 = h1->next;
    }

    while (h2 != nullptr)
    {
        listToReturn.insert(h2->value);
        h2 = h2->next;
    }

    return listToReturn;
}

int main()
{
    SLList l1, l2;
    int s1, s2;

    std::cout << "Enter size of first list : ";
    std::cin >> s1;

    for (int i = 0; i < s1; i++)
```

```
    {
        int element;
        std::cout << "Enter " << i + 1 << " element : ";
        std::cin >> element;
        l1.insert(element);
    }

    std::cout << "Enter size of second list : ";
    std::cin >> s2;

    for (int i = 0; i < s2; i++)
    {
        int element;
        std::cout << "Enter " << i + 1 << " element : ";
        std::cin >> element;
        l2.insert(element);
    }

    std::cout << std::endl
              << "First list is : " << std::endl;
    l1.print();

    std::cout << std::endl
              << "Second list is : " << std::endl;
    l2.print();

    SLList merged = mergeLists(l1.head, l2.head);

    std::cout << std::endl
```

```
            << "Merged list is : " << std::endl;
    merged.print();

    return 0;
}
```

Output Screenshots

```
Enter size of first list : 3
Enter 1 element : 1
Enter 2 element : 2
Enter 3 element : 5
Enter size of second list : 4
Enter 1 element : 0
Enter 2 element : 5
Enter 3 element : 7
Enter 4 element : 11

First list is :
1 2 5
Second list is :
0 5 7 11
Merged list is :
0 1 2 5 7 11
```

```
Enter size of first list : 3
Enter 1 element : 1
Enter 2 element : 4
Enter 3 element : 10
Enter size of second list : 2
Enter 1 element : 2
Enter 2 element : 7

First list is :
1 4 10
Second list is :
2 7
Merged list is :
1 2 4 7 10
```