Lab Assignment 1

10-10-2022 22MCA0139 RAJAT SINGH

ITA5002 - Problem Solving with Data Structures and Algorithms Lab

1) 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.

Solution:

```
#include<iostream>
#include<string>
using namespace std;

char stack[1];
int top=-1,n=1;

void push(char a) {
    top++;
    stack[0]=a;
}

void pop(){
    top--;
}

int main(){

    //char str[16];
    string str;

    cout<<"Enter the 15 bits : "<<endl;
    getline(cin, str);

    if(str.length()!=15){
        cout<<"Invalid Input!"<<endl;
        return 0;
    }

    else{
        for(int i=0; i<15;i++){
            if(str[i]=='1'){
                if(top==-1)
                    push('1');
```

```
                else
                    pop();
            }
            else if(str[i]=='0')
                continue;
            else{
                cout<<"Invalid Input!"<<endl;
                return 0;
            }
        }

        if(top==-1){
            cout<<"The parity is even : "<<endl;
            str=str+'0';
            cout<<str;
        }
        else{
            cout<<"The parity is odd : "<<endl;
            str=str+'1';
            cout<<str;
            }
    }
    return 0;
}
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

101100011101001
The parity is even :
1011000111010010
D:\Backup_28Sep\Study Material\MCA\dsa questions\strings [main ↑2 +3 ~2 -0 !]> .\a
Enter the 15 bits :
101100011101011
The parity is odd :
1011000111010111
D:\Backup_28Sep\Study Material\MCA\dsa questions\strings [main ↑2 +3 ~2 -0 !]> .\a
Enter the 15 bits :
101170011101011
Invalid Input!
D:\Backup_28Sep\Study Material\MCA\dsa questions\strings [main ↑2 +3 ~2 -0 !]> .\a
Enter the 15 bits :
25a
Invalid Input!
D:\Backup_28Sep\Study Material\MCA\dsa questions\strings [main ↑2 +3 ~2 -0 !]>
```

3. Implement a program to perform the Polynomial Addition using Singly Liked List.

Solution:

```
#include <iostream>
using namespace std;

struct Node {
    int num;
    int power;
    struct Node* next;
};

void createNode(int x, int y, struct Node** temp)
{
    struct Node *a, *b;
    b = *temp;
    if (b != NULL) {
        a->num = x;
        a->power = y;
        a->next = (struct Node*)malloc(sizeof(struct Node));
        a = a->next;
        a->next = NULL;
    }
    else {
        a = (struct Node*)malloc(sizeof(struct Node));
        a->num = x;
        a->power = y;
        *temp = a;
        a->next = (struct Node*)malloc(sizeof(struct Node));
        a = a->next;
        a->next = NULL;
    }
}

void additionPoly(struct Node* P1, struct Node* P2,
             struct Node* ans)
{
    while (P1->next && P2->next) {
        if (P1->power > P2->power) {
            ans->power = P1->power;
            ans->num = P1->num;
            P1 = P1->next;
        }

        else if (P1->power < P2->power) {
            ans->power = P2->power;
            ans->num = P2->num;
            P2 = P2->next;
```

```
        }

        else {
            ans->power = P1->power;
            ans->num = P1->num + P2->num;
            P1 = P1->next;
            P2 = P2->next;
        }

        ans->next
            = (struct Node*)malloc(sizeof(struct Node));
        ans = ans->next;
        ans->next = NULL;
    }
    while (P1->next || P2->next) {
        if (P1->next) {
            ans->power = P1->power;
            ans->num = P1->num;
            P1 = P1->next;
        }
        if (P2->next) {
            ans->power = P2->power;
            ans->num = P2->num;
            P2 = P2->next;
        }
        ans->next
            = (struct Node*)malloc(sizeof(struct Node));
        ans = ans->next;
        ans->next = NULL;
    }
}

void displayPoly(struct Node* node)
{
    while (node->next != NULL) {
        printf("%dx^%d", node->num, node->power);
        node = node->next;
        if (node->num >= 0) {
            if (node->next != NULL)
                printf("+");
        }
    }
}

int main()
{
    struct Node *P1 = NULL, *P2 = NULL, *ans = NULL;
```

```
    printf("1st Polynomial: ");
    createNode(6, 2, &P1);
    createNode(5, 1, &P1);
    createNode(3, 0, &P1);
    displayPoly(P1);

    printf("\n2nd Polynomial: ");
    createNode(-3, 1, &P2);
    createNode(-2, 0, &P2);
    displayPoly(P2);

    ans = (struct Node*)malloc(sizeof(struct Node));

    additionPoly(P1, P2, ans);

    printf("\nAdded Polynomial: ");
    displayPoly(ans);

    return 0;
}
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

D:\Backup_28Sep\Study Material\MCA\dsa questions\linkedList [main ↑2]> g++ .\addTwoPolynomials.cpp
D:\Backup_28Sep\Study Material\MCA\dsa questions\linkedList [main ↑2 +0 ~1 -0 !]> .\a
1st Polynomial: 6x^2+5x^1+3x^0
2nd Polynomial: -3x^1-2x^0
Added Polynomial: 6x^2+2x^1+1x^0
D:\Backup_28Sep\Study Material\MCA\dsa questions\linkedList [main ↑2 +0 ~1 -0 !]>
```

4. 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.

Solution:

```
#include<bits/stdc++.h>
#include<iostream>
using namespace std;

class Node
{
    public:
    int data;
    Node* next;
};

void push(Node** head_ref, int new_data)
{
    Node* new_node = new Node();

    new_node->data = new_data;

    new_node->next = (*head_ref);

    (*head_ref) = new_node;
}

void printList(Node *node)
{
    while (node!=NULL)
    {
        cout<<node->data<<" ";
        node = node->next;
    }
}

Node* mergeSLL(struct Node* a, struct Node* b)
{
    Node* result = NULL;

    if (a == NULL)
        return (b);
    else if (b == NULL)
        return (a);

    if (a->data <= b->data) {
        result = a;
        result->next = mergeSLL(a->next, b);
    }
```

```
    else {
        result = b;
        result->next = mergeSLL(a, b->next);
    }
    return (result);
}

void removeDuplicates(Node* head)
{
    Node* current = head;

    Node* next_next;

    if (current == NULL)
        return;

    while (current->next != NULL) {

        if (current->data == current->next->data) {

            next_next = current->next->next;
            free(current->next);
            current->next = next_next;
        }
        else
        {
            current = current->next;
        }
    }
}

Node* mergeWithoutDuplicates(Node* head1, Node* head2)
{
    Node* head = mergeSLL(head1, head2);

    removeDuplicates(head);

    return head;
}

int main()
{
    Node* res = NULL;
    Node* a = NULL;
    Node* b = NULL;

    push(&a, 15);
    push(&a, 10);
```

```
    push(&a, 10);
    push(&a, 6);
    push(&a, 5);

    push(&b, 20);
    push(&b, 6);
    push(&b, 6);
    push(&b, 4);
    push(&b, 2);

    res = mergeWithoutDuplicates(a, b);

    cout << "Merged Linked List is: \n";
    printList(res);

    return 0;
}
```

Output: