WEEK – 2

CODE HACKATHON

(Department-CSE)

Date: 4th August 2018 Total Marks: **100**

Created by: - Nishanth ¾ B-16 – 7093197160

V.Srinivas ¾ B-16 – 8374256375

Ajit Panigrahi ¾ B-12 - 8895567749

M.Pranay ¾ B-16 - 8919345427

Q1) Write the program to find the minimum spanning tree for the following graph:

(**15-MARKS**)

```
      10
        0--------1
        |  \     |
       6|   5\   |15
        |      \ |
        2--------3
           4
```

Sample output:

```
Following are the edges in the constructed MST
2 -- 3 == 4
0 -- 3 == 5
0 -- 1 == 10
```

Solution:

```
// C++ program for Kruskal's algorithm to find Minimum Spanning Tree
// of a given connected, undirected and weighted graph
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// a structure to represent a weighted edge in graph
struct Edge
{
    int src, dest, weight;
};

// a structure to represent a connected, undirected
// and weighted graph
struct Graph
{
    // V-> Number of vertices, E-> Number of edges
    int V, E;

    // graph is represented as an array of edges.
    // Since the graph is undirected, the edge
    // from src to dest is also edge from dest
    // to src. Both are counted as 1 edge here.
    struct Edge* edge;
};

// Creates a graph with V vertices and E edges
struct Graph* createGraph(int V, int E)
{
    struct Graph* graph = new Graph;
    graph->V = V;
    graph->E = E;

    graph->edge = new Edge[E];

    return graph;
}

// A structure to represent a subset for union-find
struct subset
{
    int parent;
    int rank;
};

// A utility function to find set of an element i
// (uses path compression technique)
int find(struct subset subsets[], int i)
{
    // find root and make root as parent of i
    // (path compression)
    if (subsets[i].parent != i)
        subsets[i].parent = find(subsets, subsets[i].parent);

    return subsets[i].parent;
}

// A function that does union of two sets of x and y
// (uses union by rank)
void Union(struct subset subsets[], int x, int y)
{
    int xroot = find(subsets, x);
    int yroot = find(subsets, y);

    // Attach smaller rank tree under root of high
    // rank tree (Union by Rank)
    if (subsets[xroot].rank < subsets[yroot].rank)
        subsets[xroot].parent = yroot;
    else if (subsets[xroot].rank > subsets[yroot].rank)
        subsets[yroot].parent = xroot;
    // If ranks are same, then make one as root and
    // increment its rank by one
    else
    {
        subsets[yroot].parent = xroot;
        subsets[xroot].rank++;
    }
}

// Compare two edges according to their weights.
// Used in qsort() for sorting an array of edges
int myComp(const void* a, const void* b)
{
    struct Edge* a1 = (struct Edge*)a;
    struct Edge* b1 = (struct Edge*)b;
    return a1->weight > b1->weight;
}

// The main function to construct MST using Kruskal's algorithm
void KruskalMST(struct Graph* graph)
{
    int V = graph->V;
    struct Edge result[V];  // Tnis will store the resultant MST
    int e = 0;  // An index variable, used for result[]
    int i = 0;  // An index variable, used for sorted edges

    // Step 1:  Sort all the edges in non-decreasing
    // order of their weight. If we are not allowed to
    // change the given graph, we can create a copy of
    // array of edges
    qsort(graph->edge, graph->E, sizeof(graph->edge[0]), myComp);

    // Allocate memory for creating V ssubsets
    struct subset *subsets =
        (struct subset*) malloc( V * sizeof(struct subset) );

    // Create V subsets with single elements
    for (int v = 0; v < V; ++v)
    {
        subsets[v].parent = v;
        subsets[v].rank = 0;
    }

    // Number of edges to be taken is equal to V-1
    while (e < V - 1)
    {
        // Step 2: Pick the smallest edge. And increment
        // the index for next iteration
        struct Edge next_edge = graph->edge[i++];

        int x = find(subsets, next_edge.src);
        int y = find(subsets, next_edge.dest);

        // If including this edge does't cause cycle,
        // include it in result and increment the index
        // of result for next edge
        if (x != y)
        {
            result[e++] = next_edge;
            Union(subsets, x, y);
        }
        // Else discard the next_edge
    }

    // print the contents of result[] to display the
    // built MST
    printf("Following are the edges in the constructed MST\n");
    for (i = 0; i < e; ++i)
        printf("%d -- %d == %d\n", result[i].src, result[i].dest,
                                                 result[i].weight);
    return;
}

// Driver program to test above functions
int main()
{
    int V = 4;  // Number of vertices in graph
    int E = 5;  // Number of edges in graph
    struct Graph* graph = createGraph(V, E);


    // add edge 0-1
    graph->edge[0].src = 0;
    graph->edge[0].dest = 1;
    graph->edge[0].weight = 10;

    // add edge 0-2
    graph->edge[1].src = 0;
    graph->edge[1].dest = 2;
    graph->edge[1].weight = 6;

    // add edge 0-3
    graph->edge[2].src = 0;
    graph->edge[2].dest = 3;
    graph->edge[2].weight = 5;

    // add edge 1-3
    graph->edge[3].src = 1;
    graph->edge[3].dest = 3;
    graph->edge[3].weight = 15;

    // add edge 2-3
    graph->edge[4].src = 2;
    graph->edge[4].dest = 3;
    graph->edge[4].weight = 4;

    KruskalMST(graph);
    return 0;
}
```

Q2) Speak like Yoda from Star Wars, where "I am Pranay" becomes, "Pranay, I am."

[For the input line, shift the last word to the first and add a comma & space.]

(**10-MARKS**)

Input:

We play monopoly.

Output:
Monopoly, we play.

[Marks for: capitalize ‘monopoly’, lowercase ‘we’, full stop placement]

Solution:

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

int main() {
        string input = "We play Monopoly.";
        // string input = "I'm known as Pranay.";
        // string input = "Hello world!";

        // Take command line input here:
        // string input;
        // cin >> input;

        int len = input.length();
        if (input[len - 1] != '\0') {
                input += '\0';
        }

        string ans = "";
        int lastWordPos = 0;
        for (int i = 0; i < len; ++i) {
                if (input[i] == ' ') {
                        lastWordPos = i;
                }
        }

        for (int i = lastWordPos + 1; i < len - 1; ++i) {
                ans += input[i];
        }
        // Capitalize first letter
        if (ans[0] >= 'a' && ans[0] <= 'z') {
                ans[0] -= 32;
        }

        ans += ", ";

        // Convert uppercase letters to lowercase, while ignoring "I
am...", "I'm..."
        if (!(input[0] == 'I' && (input[1] == ' ' || input[1] == '\'')) &&
input[0] >= 'A' && input[0] <= 'Z') {
                ans += input[0] + 32;
        }
        else {
                ans += input[0];
        }

        for (int i = 1; i < lastWordPos; ++i) {
                ans += input[i];
        }
        ans += '.';

        cout << ans << endl;
        return 0;
}
```

Q3)You are given a function - Node* mergelists(Node* head1,Node* head2).This function takes two head pointers of two different SORTED Linked lists.(10-MARKS)

Given these two head pointers, MANIPULATE THE LINKS between the two SORTED lists such that it forms a Merged List that is still SORTED.

Return the head pointer that points to the third SORTED list after manipulating the list via the function "mergelists".

YOU ONLY HAVE TO RETURN THE POINTER , THE OUTPUT IS ALREADY MANAGED.

COMMENT OUT THE FUNCTION "mergelists" AND COMPLETE IT ACCORDING TO THE TASK.

SAMPLE CASE:

List 1:

1->2->3->NULL

List 2:

3->4->NULL

OUTPUT LIST:

1->2->3->3->4-NULL

OUTPUT LIST STRUCTURE:

```
1->2->3-|
        |->3->4->NULL.
```

IMPORTANT NOTE:Either of the head pointer passed to the function head1 or head2 may be NULL.

*/

Solution:

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

class Node
{
public:

                int data;
                Node* next;
        Node(int n)
        {
                this->data=n;
                this->next=NULL;
        }


};

class List
{
public:

        Node* head;
        Node* tail;
        List()
        {
                this->head=NULL;
                this->tail=NULL;
        }

        void insert_node(int data)
        {
                Node* n=new Node(data);
                if(this->head==NULL)
                {
                        this->head=n;
                }
                else
                {
                        this->tail->next=n;
                }
                this->tail=n;
        }
        void display_list()
        {
                if(head==NULL)
                    cout<<"EMPTY"<<endl;
                Node* temp=head;
                while(temp)
                {
                        cout<<temp->data<<" ";
                        temp=temp->next;
                }
                cout<<endl;
        }
};

Node* mergelists(Node* head1,Node *head2)//Complete this
function..........
{
    if(head1==NULL)//List 1 is empty.....
        return(head2);

    else if(head2==NULL)//List 2 is empty.....
        return(head1);

    Node* temp1=head1;//Pointer to first list.
    Node* temp2=head2;//Pointer to second list.

    //Our third list...........................
    Node* head3;
    Node* temp3;
    //First compare heads of the lists.........
    if(head1->data<=head2->data)//Head of third list will point to first
list.
    {
        head3=head1;
        temp1=temp1->next;      //Now temp is at second node.
    }
    else                        //Head of third list will point to second
list.
    {
        head3=head2;
        temp2=temp2->next;      //Now temp is at second node.
    }

    //Now initialize temp3 to the head of third list head3.
    temp3=head3;
    while(temp1!=NULL && temp2!=NULL)
    {
        if(temp1->data<=temp2->data)//Add the node from first list to
list3.
        {
            temp3->next=temp1;
            temp1=temp1->next;
        }
        else                        //Add the node from second list to
list3.
        {
            temp3->next=temp2;
            temp2=temp2->next;
        }
        temp3=temp3->next;          //In either case after adding move the
pointer in list3.
    }
    if(temp1==NULL)                 //If list one is exhausted, add
remaining part of list2.
    {
        temp3->next=temp2;
        temp2=temp2->next;
    }
    else                            //If list two is exhausted, add
remaining part of list1.
    {
        temp3->next=temp1;
        temp1=temp1->next;
    }
    return(head3);                  //Return the head of the new list.
}
//.................................................................
.

main()
{

        int n1,n2;
        int value;
        cout<<"Enter the number of nodes in the first list:";
        cin>>n1;
        cout<<"Enter the number of nodes in the second list:";
        cin>>n2;
        List* l1=new List();
        List* l2=new List();
        List* l3=new List();
        //Input List 1......................
        if(n1>0)
        {
            cout<<"Enter the first list:";
            for(int i=0;i<n1;i++)
            {
                    cin>>value;
                    l1->insert_node(value);
            }
        }

        //Input List 2.......................
        if(n2>0)
        {
            cout<<"Enter the second list:";
            for(int i=0;i<n2;i++)
            {
                    cin>>value;
                    l2->insert_node(value);
            }
        }
        //..................................

        //Displaying the Initial Lists........
        cout<<"List 1:"<<endl;
        l1->display_list();
        cout<<endl;
        cout<<"List 2:"<<endl;
        l2->display_list();
        cout<<endl;

        //Output............................
        cout<<"Output:"<<endl;
        l3->head=mergelists(l1->head,l2->head);
        l3->display_list();
        //..................................
}
```

Q4) A **'C'** Example to use 'kbhit' function:

(**10-MARKS**)

Sample output: "Press a key" will keep printing on the

Console until the user presses a key on the keyboard.

Solution:

```
#include<stdio.h>
#include <conio.h>

int main()
```

```
{
        do
        {
                printf("Press any key to stop loop.\n");
        } while (!kbhit());

        return 0;
}
```

Q5) There are N couples in cinema hall and one more person who came alone. For given 2*N+1 input elements, find that unique individual if all of the couples are denoted by the same number.

(**5-MARKS**)

Input:

88 120 34 2 345 3 2 90 88 120 3 90 34

Output:

345

Solution:

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

int main() {
        int N;
        cin >> N;
        int all[N];
        for (int i = 0; i < N; ++i) {
                cin >> all[i];
        }
        int ans = 0;
        for (int i = 0; i < N; ++i) {
                ans ^= all[i]; // XOR operation
                // For some k,
                // k XOR k = 0
                // 0 XOR k = k
                //
                // So, XOR of duplicates gives 0, the remaining one being
the answer
        }
        cout << ans << endl;
        return 0;
}
```

Q6) You are given two Cars on a number line ready to move in the positive direction (i.e. toward positive infinity).
-The first Car starts at location x1 and moves at a rate of v1 meters per unit time.

-The second Car starts at location x2 and moves at a rate of v2 meters per time.

You have to figure if we can get both Cars at the same location at the same time. If it is possible, return "YES", otherwise return "NO".

(**10-MARKS**)

INPUT:
A single line of four space-separated integers denoting the respective values of x1,v2,x2,v2.

OUTPUT:
YES if they will meet.
NO if they won't.

Example:
Car1:
(Position,Velocity)=(0,3)
Car2:
(Position,Velocity)=(4,2)

After 4 Seconds:
Position of Car1=0+(4*3)=12
Position of Car2=4+(4*2)=12

They meet therfore we print Yes.

Solution:

```
string cars_meet(int x1, int v1, int x2, int v2) {
    if((v1<v2) && (x1<x2))
    {
        return("NO");
    }
    else{
        if((v1!=v2)&&((x2-x1)%(v1-v2))==0)
            return("YES");
        else
        {
            return("NO");
        }
    }
}
```

Q7)Program to print weekday of given date (10 MARKS)

Sample output:
First Run:

```
    Input date (DD-MM-YYYY): 22-02-2015
    Date is correct [22/02/2015].
    Week day is: Sunday
    Second Run:
    Input date (DD-MM-YYYY): 01-01-2016
    Date is correct [01/01/2016].
    Week day is: Friday

    Third Run:
    Input date (DD-MM-YYYY): 30-02-2010
    Date is in-correct.

    Fourth Run:
    Input date (DD-MM-YYYY): 29-02-2001
    Date is in-correct.

    Fifth Run:
    Input date (DD-MM-YYYY): 30-13-2011
    Date is in-correct.
```

Solution:

```
/*C program to validate date and print weekday of given date.*/
#include <stdio.h>

/*
 * function: validateDate
 * arguments: d- day, m- month, y- year
 * return type: 0 - invalid, 1 - valid
 * */
int validateDate(int d,int m,int y)
{
    //check year validity
    if(y>=1800 && y<=2999)
    {
        //check month validity
        if(m>=1 && m<=12)
        {
            //check day validity
            if(d>=1 && d<=31)
            {
                if( (d>=1 && d<=30) && (m==4||m==6||m==9||m==11))
                    return 1;   //valid date
                else if((d>=1 && d<=30) &&
(m==1||m==3||m==5||m==7||m==8||m==10||m==12))
                    return 1;   //valid date
                else if((d>=1 && d<=28) && (m==2))
                    return 1;   //valid date
                else if(d==29 && m==2 && ((y%400==0)||(y%4==0 && y%4!=0)))
                    return 1;   //valid date
                else
                    return 0;   //invalid day
            }
            else
            {
                return 0;   //day is invalid
            }
        }
        else
        {
            return 0; //month is invalid
        }
    }
    else
    {
        return 0; //year is invalid
    }
}
// This function will return week day number from 0 to 6
int wd(int year, int month, int day)
{
    int wday=0;
    wday=(day  + ((153 * (month + 12 * ((14 - month) / 12) - 3) + 2) / 5)
\
        + (365 * (year + 4800 - ((14 - month) / 12)))              \
        + ((year + 4800 - ((14 - month) / 12)) / 4)                \
        - ((year + 4800 - ((14 - month) / 12)) / 100)              \
        + ((year + 4800 - ((14 - month) / 12)) / 400)              \
        - 32045                                                    \
      )%7;
     return wday;
}

int main()
{
    int day,month,year;
    int wDayNo=0;
    char
dayNames[][12]={"Monday","Tuesday","Wednesday","Thursday","Friday","Saturd
ay","Sunday"};

    //input date
    printf("Input date (DD-MM-YYYY): ");
    scanf("%d-%d-%d",&day,&month,&year);

    //check date is correct or not
    if(validateDate(day,month,year)==1){
        printf("Date is correct [%02d/%02d/%02d].\n",day,month,year);
        //get weekday number
        wDayNo=wd(year,month,day);
        //print weekday according to wDayNo
        printf("week day is: %s\n",dayNames[wDayNo]);
    }
    else
        printf("Date is in-correct.\n");
    return 0;
```

Q8)Nishant plays with a list of numbers and decides to cycle its elements in alternate directions in the Fibonacci series. First he shifts every element one step forward, the last element moved to the first, then he shifts every element one step backwards, moving the first element to the back. Then he shifts elements by 2 positions forward, and then 3 positions back. For a given string, produce the final string after Nth shift in the Fibonacci series.

(15 MARKS)

Input:

```
Nishant
5
```

Output:

```
hantNis
```

[+1, -1, +2, -3, +5]

Solution:

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

int main() {
        string input;
        // cin >> input;
        input = "Nishanth";

        int len = input.length();

        int N; // minimum 2
        // cin >> N;
        N = 5;

        int fibo[N];
        fibo[0] = 1;
        fibo[1] = 1;
        for (int i = 2; i < N; ++i) {
                fibo[i] = fibo[i - 1] + fibo[i - 2];
        }

        int rotateBy = 0;
        for (int i = 0; i < N; ++i) {
                if (i & 1 == 1) {
                        fibo[i] *= (-1);
                }
                rotateBy += fibo[i];
        }

        if (rotateBy < 0) {
                rotateBy = -((-rotateBy) % len);
                // OR simply:
                // rotateBy *= (-1);
                // rotateBy %= len;
                // rotateBy *= (-1);
        }

        string ans = "";
        for (int i = -rotateBy; i < len - rotateBy; ++i) {
                ans += (i < 0) ? input[len + i] : input[i];
        }

        cout << ans << endl;
        return 0;
}
```

Q9) A 'C' program to find Binary Addition and Binary Subtraction (take the numbers in decimal format and print them in decimal format) (5-MARKS)

Sample output:

```
Input first integer value: 30
Input second integer value: 5
Binary Addition: 35
Binary Subtraction: 25
```

Solution:

```
#include <stdio.h>

//function for Binary Addition
int binAddition(int a,int b)
{
      int c; //carry
      while (b != 0) {
              //find carry and shift it left
              c = (a & b) << 1;
              //find the sum
              a=a^b;
              b=c;
      }
      return a;
}

//function for Binary Subtraction
int binSubtracton(int a, int b)
{
      int carry;
      //get 2's compliment of b and add in a
      b = binAddition(~b, 1);

      while (b != 0) {
              //find carry and shift it left
              carry = (a & b) << 1;
              //find the sum
              a = a ^ b;
              b = carry;
      }
      return a;
}


int main()
{
    int number1,number2, binAdd, binSub;

    printf("Input first integer value: ");
    scanf("%d",&number1);

    printf("Input second integer value: ");
    scanf("%d",&number2);

    binAdd=binAddition(number1,number2);
    binSub=binSubtracton(number1,number2);

    printf("Binary Addition: %d\n",binAdd);
    printf("Binary Subtraction: %d\n",binSub);

    return 0;

}
```

Q10)A palindrome is a word, phrase, number, or other sequence of characters which reads the same backwards and forwards.
Determine if a given string - S is a palindrome.

To solve this challenge, we must first take each character in , enqueue it in a queue, and also push that same character onto a stack.
Once that's done, we must dequeue the first character from the queue and pop the top character off the stack, then compare the two characters to see if they are the same;
As long as the characters match, we continue dequeuing, popping, and comparing each character until our containers are empty (a non-match means it isn’t a palindrome).

Complete the methods below as asked. (10-MARKS)

Sample Input:
aaa

Sample Output:
PALINDROME

Solution:

```
#include <iostream>
#include<stdlib.h>
#include<string.h>
using namespace std;
char a[100];//A Stack represented by a Global Array.
char b[100];//A Queue represented by a Global Array.
```

```
int i=0,j=0,k=0;//Three variables index the Data structures. You may use
them as you like or define your own variables instead of these.

void pushCharacter(char ch)//Adds a character to the Stack.
    {
        i++;//First Increment the top as we have no previous top.
        a[i]=ch;//The new top is our added character.
    }
void enqueueCharacter(char ch)//Adds a character to Queue.
{
        b[k++]=ch;//Just adds to position k in queue and increments the
end part of queue-k after adding to further store it.
}
char popCharachter()
{
    char temp=a[i];//Hold the top character in stack.
    i--;//Reduce top as it is now considered popped.
    return(temp);//Return the previous topmost character.
}

char dequeCharachter()
{
    char temp=b[j];//Take char from the front of the queue.
    int p;
    //Shift all elements left by one position..
    for(p=j;p<k;p++)
        b[p]=b[p+1];
    return(temp);//Return that value
}
main(){
    string str;
    cout<<"Enter string:";
    cin>>str;
    for(int i=0;i<str.length();i++)
    {
        pushCharacter(str[i]);
        enqueueCharacter(str[i]);
    }
    bool same=true;
    for(int i=str.length();i>=0;i--)
    {
        same=same&&(popCharachter()==dequeCharachter());
    }
    if(same)
        cout<<"PALINDROME"<<endl;
    else
        cout<<"NOT PALNDROME"<<endl;
};
```

-----------------------------The End-------------------------------------

All the best