Faculty of Computers and Artificial intelligence-
Cairo University (credit hours system)

Cairo University, Faculty of
Computers and Artificial intelligence

Final Year Project 2020

Part II

Subject: Data Structures

Subject code: SCS214

Project ID: PM-453

Under the supervision of: Dr. Basheer Abdel Fatah

TA. Esraa Salem

ID	Name	الأسم	Group
20186031	Nada Mohamed	ندي محمد	3
20186007	Ayat Hany	أيات هاني	
20186008	Sarah Khaled	سارة خالد	
20186043	Mark Rofaeel	مارك روفائيل	

A. Student management system:

```
//===========================================================================
// Name        : Student management system.cpp
// Author      : Nada Mohamed - Ayat Hany - Sarah Khaled - Mark Rofaeel
// ID          : 20186031     - 20186007  - 20186008     - 20186043
// Version     : 1.0
// Copyright   : Your copyright notice
// Description : Linked lists
//===========================================================================

#include <iostream>
using namespace std;

class Course
{
       friend class CourseList;
    protected:
              string subjectname;
              int total;
              char grade;
              float point;
       public:
              Course()
              {
                     subjectname = "";
                     total = 0;
                     grade = '\0';
                     point = 0.0;
              }
              Course(string name,int t,char c,float p)
              {
                     subjectname = name ;
                     total = t ;
                     grade = c;
                     point = p;
              }
};
class CourseList
{
       friend class Course;
    struct node
       {
              string name;
              int t;
              char g;
              float p;
              node* next;
    };
    node*first;
    node*last;
    int length;
       public:
              Course *s;
              CourseList()
           {
                  first = last = NULL;
```

```
                length = 0;
            }
               void Add(Course *n)
               {
               node* NewNode = new node;
               NewNode->name = n->subjectname;
                      NewNode->t = n->total;
               NewNode->g = n->grade;
               NewNode->p = n->point;
               node* Node1 = first;
               node** Node2 = &first;
               while(Node1 != NULL && Node1->name < NewNode->name)
               {
                      Node2 = &Node1->next;
                      Node1 = Node1->next;
             }
               *Node2 = NewNode;
               NewNode->next = Node1;
        }
        void print()
               {
                      node*curr = first;
                      while(curr!=NULL)
                      {
                          cout<<curr->name<<"  ";
                         cout<<curr->t<<"  ";
                         cout<<curr->g<<"  ";
                         cout<<curr->p<<endl;
                         curr=curr->next;
                      }
               }
};
class Student
{
        friend class StudentList;
    protected:
               string name;
               string dep;
               int no_of_courses;
               CourseList*co;
        public:
               Student()
               {
                      name = "";
                      dep = "";
                      no_of_courses = 0;
               }
               Student(string na,string d,int n,CourseList*c)
               {
                      name = na;
                      dep = d;
                      no_of_courses = n;
                co = c;
               }
};
```

```
class StudentList
{
       friend class Student;
    struct node
       {
              string na;
              string de;
              int no;
              CourseList* l;
              node* next;
    };
    node*first;
    node*last;
    int length;
       public:
              StudentList()
              {
                  first = last = NULL;
                  length = 0;
       }
              void Add(Student *n)
              {
                     node* NewNode = new node;
              NewNode->na = n->name;
              NewNode->de = n->dep;
              NewNode->no = n->no_of_courses;
              NewNode->l=n->co;
              node* Node1 = first;
              node** Node2 = &first;
              while(Node1 != NULL && Node1->na < NewNode->na)
              {
                     Node2 = &Node1->next;
                     Node1 = Node1->next;
              }
              *Node2 = NewNode;
              NewNode->next = Node1;
       }
       void print()
              {
                     node*curr = first;
                     while(curr!=NULL)
                     {
                         cout<<"Student: ";
                         cout<<curr->na<<"  ";
                            cout<<curr->de<<"  ";
                            cout<<curr->no<<endl<<"Student Courses: "<<endl;
                            curr->l->print();
                            cout<<endl;
                            curr=curr->next;
                     }
              }
};
```

```
int main()
{
	Course c("Software Requirement Analysis",50,'D',2);
	Course c1("Database systems",100,'A',4);
	Course c2("Maths 1",90,'A',4);
	Course c3("Project management",60,'C',2.7);
	CourseList l;
	l.Add(&c);
	l.Add(&c1);
	l.Add(&c2);
	l.Add(&c3);
	Course h("Database systems",88,'A',3.7);
	Course h1("Data structures",100,'A',4);
	Course h2("Technical writing",40,'F',1);
	Course h3("Maths 3",70,'B',3.2);
	CourseList l2;
	l2.Add(&h);
	l2.Add(&h1);
	l2.Add(&h2);
	l2.Add(&h3);
	Student s("Sarah Khaled","CS",3,&l);
	Student s2("Nada Mohamed","IT",4,&l2);
	Student s3("Ayat Hany","CS",1,&l);
	Student s4("Mark Rofaeel","DS",2,&l2);
	StudentList list;
	list.Add(&s);
	list.Add(&s2);
	list.Add(&s3);
	list.Add(&s4);
	list.print();
	return 0;
}
```

Output:

```
D:\Documents\FCI\2019-2020\Second term\Data Structures\Project\SCS214-20186031-20186007-20186008-20186043\SCS214-20186031-20186007-20186008-20186043\Part II\A\Student management system.exe
Student: Ayat Hany  CS  1
Student Courses:
Database systems  100  A  4
Maths 1  90  A  4
Project management  60  C  2.7
Software Requirement Analysis  50  D  2

Student: Mark Rofaeel  DS  2
Student Courses:
Data structures  100  A  4
Database systems  88  A  3.7
Maths 3  70  B  3.2
Technical writing  40  F  1

Student: Nada Mohamed  IT  4
Student Courses:
Data structures  100  A  4
Database systems  88  A  3.7
Maths 3  70  B  3.2
Technical writing  40  F  1

Student: Sarah Khaled  CS  3
Student Courses:
Database systems  100  A  4
Maths 1  90  A  4
Project management  60  C  2.7
Software Requirement Analysis  50  D  2


--------------------------------
Process exited after 0.06179 seconds with return value 0
Press any key to continue . . .
```

B.I. In-place merge two sorted arrays:

```
//===========================================================================
// Name        : In-place merge two sorted arrays.cpp
// Author      : Nada Mohamed - Ayat Hany - Sarah Khaled - Mark Rofaeel
// ID          : 20186031     - 20186007  - 20186008     - 20186043
// Version     : 1.0
// Copyright   : Your copyright notice
// Description : Arrays
//===========================================================================

#include <iostream>
using namespace std;

void merge(int X[], int Y[], int m, int n)
{
	int temp, j;
	for (int i = 0; i < m;  i++)
	{
		if (X[i] > Y[0])
		{
			swap(X[i], Y[0]);
			temp = Y[0];
			for (j = 1; j < n && Y[j] < temp; j++)
			{
				Y[j - 1] = Y[j];
			}
			Y[j - 1] = temp;
		}
	}
}
```

```
int main()
{
	int X[] = {1,4,7,8,10};
	int Y[] = {2,3,9};
	int m,n;
	m=5;
	n=3;
	merge(X, Y, m, n);
	cout<<" X[] = {";
	for (int i = 0; i < m; i++)
	{
		cout << X[i];
		if(i<m-1)
		{
			cout<<",";
		}
	}
	cout <<"}"<<endl<<" Y[] = {";
	for (int i = 0; i < n; i++)
	{
		cout << Y[i];
		if(i<n-1)
		{
			cout<<",";
		}
	}
	cout <<"}"<<endl<<endl;
	int A[] = {6,10,12,31};
	int B[] = {5,9,11,20,27};
	int a,b;
	a=sizeof(A)/sizeof(A[0]);
	b=sizeof(B)/sizeof(B[0]);
	merge(A, B, a, b);
	cout<<" A[] = {";
	for (int i = 0; i < a; i++)
	{
		cout << A[i];
		if(i<a-1)
		{
			cout<<",";
		}
	}
	cout <<"}"<<endl<<" B[] = {";
	for (int i = 0; i < b; i++)
	{
		cout << B[i];
		if(i<b-1)
		{
			cout<<",";
		}
	}
	cout <<"}"<<endl;
	return 0;
}
```

Output:

```
D:\Documents\FCI\2019-2020\Second term\Data Structures\Project\SCS214-20186031-20186007-20186008-20186043\SCS214-20186031-20186007-20186008-20186043\Part II\B\B.I\In-place merge two sorted arrays.exe
 X[] = {1,2,3,4,7}
 Y[] = {8,9,10}

 A[] = {5,6,9,10}
 B[] = {11,12,20,27,31}

--------------------------------
Process exited after 0.02977 seconds with return value 0
Press any key to continue . . .
```

B.II. Tree Struct:

```
//===========================================================================
// Name        : Tree Struct.cpp
// Author      : Nada Mohamed - Ayat Hany - Sarah Khaled - Mark Rofaeel
// ID          : 20186031     - 20186007  - 20186008     - 20186043
// Version     : 1.0
// Copyright   : Your copyright notice
// Description : Tree
//===========================================================================

#include <iostream>
#include <cmath>
#include <queue>
using namespace std;

template<typename T>
struct Tree
{
	Tree(const T &v) : value(v), left(NULL), right(NULL) {}
	T value;
	Tree*left;
	Tree*right;
};
template<typename T>
void flip(Tree <T>*node)
{
	if (node == NULL)
		return;
	else
	{
	    Tree<T>* temp;
	    flip(node->left);
	    flip(node->right);
	    swap(node->left,node->right);
	}
}
template<typename T>
void flip()
{
	Tree <T>* node;
	flip(node);
}
template<typename T>
void largestValues(Tree <T>*root)
{
	if (!root)
		return;
	queue<Tree<T>*> q;
	int nc, max;
	q.push(root);
	while (1)
	{
		nc = q.size();
		if (nc == 0)
			break;
		max = INT_MIN;
```

```
                while (nc--)
                {
                        Tree<T>* front = q.front();
                    q.pop();
                    if (max < front->value)
                        max = front->value;
                    if (front->left)
                        q.push(front->left);
                    if (front->right)
                        q.push(front->right);
                }
                cout << max << " ";
        }
}
template<class T>
long treebranchesSum(Tree <T>*root, int val)
{
        if (root == NULL)
                return 0;
        val = (val*10 + root->value);
        if (root->left==NULL && root->right==NULL)
                return val;
        return treebranchesSum(root->left, val) + treebranchesSum(root->right, val);
}
template<class T>
long branchesSum (Tree <T>*root)
{
        return treebranchesSum(root, 0);
}
template <class T>
void PrintTreeInOrder(Tree<T>*p)
{
    if(p!=NULL)
        {
         PrintTreeInOrder(p->left);
         cout << p->value << " ";
         PrintTreeInOrder(p->right);
        }
}
template <class T>
void PrintTreePreOrder(Tree<T>*p)
{
    if(p!=NULL)
        {
         cout << p->value << " ";
         PrintTreePreOrder(p->left);
         PrintTreePreOrder(p->right);
        }
}
template <class T>
void PrintTreePostOrder(Tree<T>*p)
{
    if(p!=NULL)
        {
         PrintTreePostOrder(p->left);
         PrintTreePostOrder(p->right);
         cout << p->value << " ";
```

```
        }
}
int main()
{
        Tree<int>* t = new Tree<int>(1);
        Tree<int>* _1_left = new Tree<int>(2);
        Tree<int>* _1_right = new Tree<int>(3);
        t->left = _1_left;  t->right = _1_right;
        Tree<int>* _2_left = new Tree<int>(4);
        Tree<int>* _2_right = new Tree<int>(5);
        _1_left->left = _2_left;
        _1_left->right = _2_right;
        cout<<"Largest values are: ";
        largestValues(t);
        cout<<endl<<"Post Order: ";
        PrintTreePostOrder(t);
        cout<<endl<<"Pre Order: ";
        PrintTreePreOrder(t);
        cout<<endl<<"In Order: ";
        PrintTreeInOrder(t);
        cout<<endl;
        flip(t);
        cout<<"After flipping : "<<endl<<"Post Order: ";
        PrintTreePostOrder(t);
        cout<<endl<<"Pre Order: ";
        PrintTreePreOrder(t);
        cout<<endl<<"In Order: ";
        PrintTreeInOrder(t);
        cout<<endl;
        cout<<"Branches Sum: "<<branchesSum(t)<<endl;
        return 0;
}
```

Output:

```
D:\Documents\FCI\2019-2020\Second term\Data Structures\Project\SCS214-20186031-20186007-20186008-20186043\SCS214-20186031-20186007-20186008-20186043\Part II\B\B.II\Tree Struct.exe
Largest values are: 1 3 5
Post Order: 4 5 2 3 1
Pre Order: 1 2 4 5 3
In Order: 4 2 5 1 3
After flipping :
Post Order: 3 5 4 2 1
Pre Order: 1 3 2 5 4
In Order: 3 1 5 2 4
Branches Sum: 262

--------------------------------
Process exited after 0.02882 seconds with return value 0
Press any key to continue . . .
```

B.III. Balanced String

```
//===========================================================================
// Name        : Balanced String.cpp
// Author      : Nada Mohamed - Ayat Hany - Sarah Khaled - Mark Rofaeel
// ID          : 20186031     - 20186007  - 20186008     - 20186043
// Version     : 1.0
// Copyright   : Your copyright notice
// Description : Strings
//===========================================================================

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

bool areSame(char open , char close)
{
	if(open=='('&&close==')')
		return true;
	else if (open=='['&&close==']')
		return true;
	else if (open=='{'&&close=='}')
		return true;
	else
		return false;
}
bool areBalanced(string bract)
{
	stack<char> q;
	for(int i = 0;i<bract.length();i++)
	{
		if(bract[i]=='('||bract[i]=='['||bract[i]=='{')
			q.push(bract[i]) ;
		else if(bract[i]==')'||bract[i]=='}'||bract[i]==']')
	    {
			if(q.empty()||!areSame(q.top(),bract[i]))
			return false ;
			else
			q.pop() ;
	    }
	}
	if(q.empty())
		return true ;
	else
		return false ;
}
int main()
{
	string b="{()}[][{()()}()]";
	cout<<"exp= "<<b<<endl;
	if(areBalanced(b))
		cout<<"Balanced"<<endl ;
	else
		cout<<"Not Balanced"<<endl ;
	string a="{(])";
	cout<<"exp= "<<a<<endl;
	if(areBalanced(a))
```

```
            cout<<"Balanced"<<endl ;
        else
            cout<<"Not Balanced"<<endl ;
        return 0;
}
```

Output: