**LAB 2**

**Name:** Nguyễn Hữu Khang

**Student’s id:** 2011365

**Question 1:**

My student ID number is 2011365. Therefore:

a1 = 1

a2 = 3

a3 = 6

a4 = 5

At the end of line 8 stack s1 is empty and the illustration of stack s2 is drawn below:

|  |
| --- |
| 1 |
| 3 |
| 6 |
| 5 |

At the end of line 14 stack s2 is empty and the illustration of stack s1 is drawn below:

|  |
| --- |
| 5 |
| 6 |
| 3 |
| 1 |
| 15 |
| 6 |

At the end of line 18 stack s1 is:

|  |
| --- |
| 3 |
| 1 |
| 15 |
| 6 |

At the end of line 18 stack s2 is:

|  |
| --- |
| 6 |
| 5 |

**Question 2:**

**a)**

PushStack (s1, a1);

PushStack (s1, a2);

PushStack (s1, a3);

PushStack (s1, a4);

While (!EmptyStack (s1)) {

While(!EmptyStack(s1)) {

PopStack (s1, x);

PushStack (s3, x);

}

While(!EmptyStack(s3)) {

PopStack (s3, x);

Print(x); //Print number

}

} //while

PushStack (s1, a1\*a3);

PushStack (s1, a2\*a4);

While (!EmptyStack (s2)) {

PopStack (s2, x);

PushStack (s1, x);

} //while

PopStack (s1, x);

PushStack (s2, x);

PopStack (s1, x);

PushStack (s2, x);

**b)**

After finishing above operations stack s1 is empty.

After finishing above operations stack s2 is:

|  |
| --- |
| 6 |
| 15 |

**Question 3:**

**Pseudo Code:**

*algorithm RemoveN (ref sourceStack<Stack>, val N <data>) {*

*int size = 0;*

*Stack temp;*

*while(!EmptyStack(sourceStack)) {*

*PopStack(sourceStack,x);*

*PushStack(temp,x);*

*size+=1;*

*}*

*if(N>size) {*

*return;*

*}*

*int index = 1;*

*while(!EmptyStack(temp)) {*

*PopStack(temp,x);*

*if(N!=index) {*

*PushStack(sourceStack,x);*

*}*

*index++;*

*}*

*}*

**C++ Code: It is stated that stack library has been included in C++ program.**

void RemoveN (stack<int> &*s*, int *N*) {

    int size = 0;

    stack<int> temp;

    while(!*s*.empty()) {

        temp.push(*s*.top());

*s*.pop();

        size+=1;

    }

    if(*N*>size) {

        return;

    }

    int index = 1;

    while(!temp.empty()) {

        if(*N*!=index) {

*s*.push(temp.top());

        }

        temp.pop();

        index++;

    }

}

**Question 4:**

My student ID number is **2011365**. Therefore, after the following segment, the value of queue Q1 is empty, queue Q2 and stack S are illustrated below:

**Q2**

|  |
| --- |
| 2 |

**S**

|  |
| --- |
| 5 |
| 6 |
| 3 |
| 1 |
| 1 |

**Question 5:**

My student ID number is 2011365. Hence, a1 = 2, a2 = 1, a3 = 3, a4 =6, a5 =5

|  |  |
| --- | --- |
| **Line** | **Value** |
| 1 | a1 + a2 = 3 |
| 2 | a2 + a3 = 4 |
| 3 | a3 + a4 = 9 |
| 4 | a3 – a4 = -3 |
| 5 | a4 + a5 = 11 |
| 6 | a5 + a1 = 7 |
| 7 | a2 – a3 = -2 |
| 8 | a1 + a2 – a3 = 0 |
| 9 | a2 – a3 + a4 = 4 |
| 10 | a3 + a4 + a5 = 14 |
| 11 | a1 – a2 = 1 |
| 12 | a1 \* a2 = 2 |
| 13 | a2 \* a3 = 3 |
| 14 | a3 \* a4 = 18 |
| 15 | a1 + a2 \* a3 = 5 |
| 16 | a5 + a1 + a2 = 8 |
| 17 | a5 – a1 = 3 |
| 18 | a2 + a3 \* a4 = 19 |
| 19 | 0 |
| 20 | a3 + a4 \* a5 = 33 |

**a)** S1 and Q1 are shown below

**S1**

|  |
| --- |
| 33 |

**Q1**

|  |
| --- |
| 7 |
| 11 |
| -3 |
| 9 |
| 4 |
| 3 |
| 3 |
| 8 |
| 5 |
| 18 |
| 3 |
| 2 |
| 1 |
| 14 |
| 4 |

**b)**

|  |
| --- |
| **3** |
| **4** |
| **9** |
| **-3** |
| **11** |
| **7** |
| **-2** |
| **-2** |
| **4** |
| **14** |
| **1** |
| **2** |
| **3** |
| **18** |
| **5** |
| **8** |
| **3** |
| **19** |
| **19** |
| **33** |

**Question 6:**

#include<iostream>

#include<stack>

using namespace std;

struct *node* {

    int data;

*node*\* next = NULL;

};

class *Stack* {

    private:

*node*\* top;

    int maxSize = 0;

    int size = 0;

    public:

    Stack() {

        Creates();

    }

    Stack(int *maxSize*) {

        Creates(*maxSize*);

    }

    ~Stack() {

        Clear();

    }

    bool isEmpty() {

        if(size == 0) {

            return true;

        }

        return false;

    }

    bool isFull() {

        if(size == maxSize) {

            return true;

        }

        return false;

    }

    void Clear() {

*node*\* p = top;

        while(p!=NULL) {

*node*\* del = p;

            p = p -> next;

**delete** del;

        }

        top = NULL;

        size =0;

    }

    int Size() {

        return size;

    }

    void Creates() {

        top = NULL;

        size = 0;

        maxSize = 100;

    }

    void Creates(int *size*) {

        top = NULL;

*this* -> maxSize = *size*;

*size* = 0;

    }

    void Push(int *data*) {

        if(*this*->isFull()) {

            cout<<"Stack is full !"<<endl;

            return;

        }

*node*\* newNode = **new** *node*;

        newNode -> data = *data*;

        if(top == NULL) {

            top = newNode;

        }

        else {

            newNode -> next = top;

            top = newNode;

        }

        size +=1;

    }

    void Pop() {

        if(isEmpty()) {

            cout<<"Stack is empty !"<<endl;

            return;

        }

*node*\* pop = top;

        top = top -> next;

        pop -> next = NULL;

**delete** pop;

        size-=1;

    }

    int Top() {

        return top ->data;

    }

};

int main() {

*Stack* p;

    int i =0;

    while(i!=8) {

        p.Push(i);

        i++;

    }

    cout<<p.Size()<<endl;

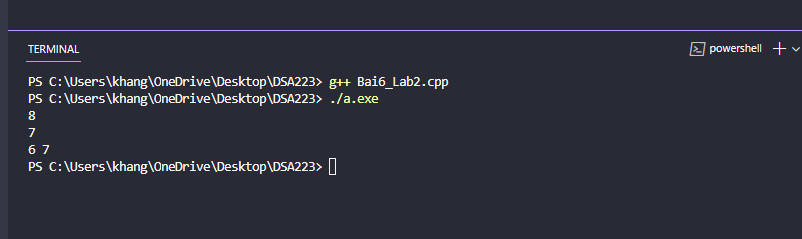
    cout<<p.Top()<<endl;

    p.Pop();

    cout<<p.Top()<<" "<<p.Size();

}

**Testing result:**

****

**Question 7:**

#include<iostream>

using namespace std;

struct *node* {

    int data;

*node*\* next = NULL;

};

class *Queue* {

    private:

*node*\* front;

*node*\* rear;

    int size = 0;

    int maxSize = 0;

    public:

    Queue() {

        Create();

    }

    Queue(int *maxSize*) {

        Create(*maxSize*);

    }

    ~Queue() {

        Clear();

    }

    void Create() {

        Clear();

        front = NULL;

        rear = NULL;

        size = 0;

        maxSize = 100;

    }

    void Create(int *maxSize*) {

        Clear();

        front = NULL;

        rear = NULL;

        size = 0;

*this* -> maxSize = *maxSize*;

    }

    bool isEmpty() {

        if (size == 0) {

            return true;

        }

        return false;

    }

    bool isFull() {

        if (size == maxSize) {

            return true;

        }

        return false;

    }

    void Clear() {

*node*\* p = front;

        while(p!=NULL) {

*node*\* del = p;

            p = p -> next;

**delete** del;

        }

        front = NULL;

        rear = NULL;

        size = 0;

    }

    int Size() {

        return size;

    }

    void EnQueue(int *data*) {

        if(isFull()) {

            cout<<"Queue is full !"<<endl;

            return;

        }

*node*\* newNode = **new** *node*;

        newNode -> data = *data*;

        if(front == NULL && rear == NULL) {

            front = newNode;

            rear = newNode;

            size ++;

        }

        else {

            rear -> next = newNode;

            rear = newNode;

            size ++;

        }

    }

    void DeQueue() {

        if(isEmpty()) {

            cout<<"Queue is empty !"<<endl;

            return;

        }

        if (size == 1) {

**delete** front;

            front = NULL;

            rear = NULL;

            size = 0;

        }

        else {

*node*\* del = front;

            front = front -> next;

**delete** del;

            size -=1;

        }

    }

    int QueueFront() {

        return front -> data;

    }

    int QueueRear() {

        return rear -> data;

    }

    void print() {

*node*\* p = front;

        while(p!=NULL) {

            cout<<p->data<<" ";

            p = p -> next;

        }

        cout<<endl;

    }

};

int main() {

*Queue* Q1;

    for (int i = 0; i<10; i++) {

        Q1.EnQueue(i);

    }

    Q1.print();

    Q1.EnQueue(10);

    Q1.print();

    Q1.DeQueue();

    Q1.print();

    cout<<Q1.QueueFront()<<endl;

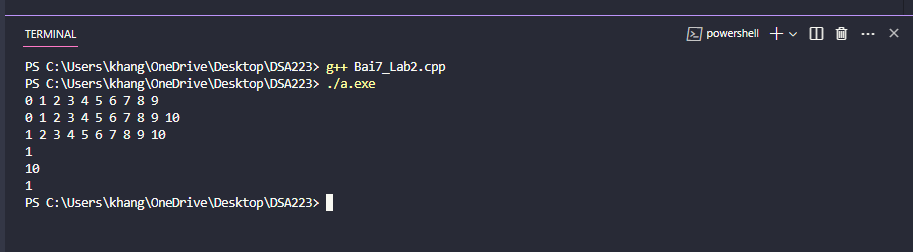
    cout<<Q1.QueueRear()<<endl;

    Q1.Clear();

    cout<<Q1.isEmpty();

}

**Testting result:**

****