Bowen Wang

301267523

CMPT 125

SUMMER 2017

**Assignment 5**

**Question 1**

**a)**

**class stack{**

**public:**

**stack();**

**stack(LinkedList\* list);**

**void push(int pushint);**

**int pop();**

**int empty();**

**private:**

**int capacity;**

**LinkedList\* stacklist;**

**};**

**b)**

**#include <iostream>**

**#include <iomanip>**

**#include "linkedlist.h"**

**using namespace std;**

**stack::stack(){**

**LinkedList\* list= new LinkedList();**

**stacklist= list;**

**capacity= 0;**

**}**

**int stack::empty(){**

**if (capacity==0){**

**return 1;**

**} else{**

**return 0;**

**}**

**}**

**void stack::push(int pushint){**

**stacklist->append(pushint);**

**capacity += 1;**

**}**

**int stack::pop(){**

**if(capacity == 0){**

**return NULL;**

**}**

**int tailnumber=stacklist->removeTail();**

**capacity -= 1;**

**return tailnumber;**

**}**

**Question 2**

**Set the pivot value as p;**

**Set len as number of elements in arr;**

**Set a new array“check” same length as arr to save the value ( p – each element in arr) ;**

**int head = 0;**

**int tail = len-1;**

**For i in range ( len ){**

**If ( check[ i ]>=0) {**

**arr[head]=p-check[i];**

**head += 1;**

**} else {**

**arr[tail] = p – check[i];**

**tail -= 1;**

**}**

**}**

**Question 3**

**void LLreverse(LL\_t\* list){**

**if (list==NULL){**

**return;**

**}**

**node\_t\* tail = list->head;**

**int i = 0;**

**while (tail->next != NULL){**

**tail = tail->next;**

**i++;**

**}**

**i++;**

**node\_t\* current= tail;**

**while (i>0){**

**node\_t\* change=list->head;**

**for (int j = 1; j<i; j++ ){**

**change= change->next;**

**}**

**current->next=change;**

**current=change;**

**i--;**

**}**

**current->next =NULL;**

**list->head=tail;**

**}**

**Question 4**

**a)**

**…**

**int i=1, j=1;**

**node\_t\* nodem=list->head;**

**node\_t\* noden=list->head;**

**while(i<n){**

**noden=noden->next;**

**i++;**

**}**

**while(j<m){**

**nodem=nodem->next;**

**j++;**

**}**

**noden->next=nodem;**

**…**

**b)**

**int LLcycle(LL\_t\* list){**

**if (list == NULL || list->head==NULL){**

**return 0;**

**}**

**node\_t\* test1=malloc(sizeof(node\_t));**

**node\_t\* test2=malloc(sizeof(node\_t));**

**test1=list->head;**

**test2=list->head;**

**while(test1->next != NULL && test2->next->next!= NULL){**

**test1=test1->next;**

**test2=test2->next->next;**

**if (test1==test2){**

**return 1;**

**}**

**}**

**return 0;**

**}**