# **Câu 1:** Hoàn thiện nốt đoạn chương trình C++ dưới đây để định nghĩa biến toàn cục lop64TH1 là một mảng tĩnh của các con trỏ tới các hồ sơ sinh viên. Sau đó, in ra số điện thoại liên lạc của sinh viên thứ 5 trong mảng. Mỗi hồ sơ sinh viên phải lưu mssv, họ và tên, địa chỉ email, số điện thoại liên lạc (đều là kiểu string).

#include <iostream>

#include <string>

#include <conio.h>

using namespace std;

class HoSoSinhVien {

public:

string mssv; // Ma so sinh vien

string hoten; // Ho va ten cua sinh vien

string email; // Dia chi Email cua sinh vien

string sdt; // So dien thoai cua sinh vien

};

#define MAX\_N\_SINH\_VIEN 100

HoSoSinhVien lop64TH1[MAX\_N\_SINH\_VIEN];

int soSinhVien = 5;

void Nhap\_Danh\_Sach\_Sinh\_Vien()

{

for (int i = 0; i < soSinhVien; i++)

{

cout << "\n\n\t\t NHAP THONG TIN SINH VIEN THU " << i + 1<<endl;

//while (getchar() != '\n');

cout << "\nNhap ma so sinh vien: ";

getline(cin, lop64TH1[i].mssv);

cout << "\nNhap ho ten sinh vien: ";

getline(cin, lop64TH1[i].hoten);

cout << "\nEmail: ";

getline(cin,lop64TH1[i].email);

cout << "\nSDT: ";

getline(cin,lop64TH1[i].sdt);

}

}

int main( )

{

Nhap\_Danh\_Sach\_Sinh\_Vien();

cout << lop64TH1[4].sdt << endl;

return 0;

}

# **Câu 2:** Cho biết đầu ra của 2 đoạn chương trình sau. (Chú ý phải viết chính xác output. Để ý các dấu cách và xuống dòng).

int lan, diep;

void f1()

{

int lan=5;

diep=9;

void f2()

{

lan ++;

diep ++;

}

int main()

{

lan = 1; diep = 2;

f1(); f2();

cout<<lan<<” ”<<diep<<endl;

f1();

cout<<lan<<” ”<<diep<<endl;

Return 0;

}

Output 1:

2 10

2 9

void nhayCoc(int buoc){

if(buoc ≤ 0){

cout<<endl;

return ;

}

cout<<” ”<<buoc;

if((buoc % 3) = = 0)

nhayCoc(buoc/3);

else

nhayCoc(buoc – 1);

cout<<” ”<<buoc;

}

int main(){

nhayCoc(14);

return 0;

}

Output 2:

14 13 12 4 3 1

1 3 4 12 13 14

# **Câu 3:** Giả sử có một lớp AwesomeQueue hoàn chỉnh để cài đặt cấu trúc dữ liệu trừu tượng hàng đợi của các ký tự, với các hàm thành viên theo đúng ý nghĩa của CTDL này:

#include <iostream>

using namespace std;

class QueueNode{

public:

char val;

QueueNode\* next;

};

class AwesomeQueue {

private:

QueueNode\* \_first;

QueueNode\* \_last;

public:

void enqueue( char c );

char dequeue( );

void PrintQueue();

};

void AwesomeQueue::enqueue( char c )

{

QueueNode\* p = new QueueNode();

p->val = c;

p->next = NULL;

if(!\_first){

\_first=\_last=p;

}

else{

\_last->next=p;

\_last=p;

}

}

char AwesomeQueue::dequeue(){

char data = \_first->val;

QueueNode\* p = \_first;

\_first = \_first->next;

delete p;

return data;

}

void AwesomeQueue::PrintQueue(){

QueueNode\* k = \_first;

while(k!=NULL){

cout<<k->val<<" ";

k=k->next;

}

}

int main(){

AwesomeQueue q;

q.enqueue('H'); q.enqueue('V'); q.enqueue('D'); q.enqueue('S');

cout << q.dequeue()<< q.dequeue() << q.dequeue() << endl;

q.enqueue('M'); q.enqueue('R'); q.enqueue('T');

cout << q.dequeue() << q.dequeue() << q.dequeue() << endl;

//q.PrintQueue();

return 0;

}

Output:

DVH

RMS

# **Câu 4:** Hoàn thiện nốt hàm bubbleSort(…) dưới đây để cài đặt thuật toán sắp xếp nổi bọt. Chúng ta cần sắp xếp một mảng các số thực (dấu phẩy động) theo thứ tự giảm dần của phần nguyên. Nếu 2 số thực có cùng phần nguyên, thứ tự giữa 2 số này trong kết quả cuối cùng không quan trọng.

#include <iostream>

using namespace std;

// Ham nhap phan tu

void input(float a[], int n){

for(int i=0; i<n; i++){

cout<<"a["<<i<<"] = ";

cin>>a[i];

}

}

// Ham xuat phan tu

void output(float a[], int n){

cout<<"Mang a: ";

for(int i=0; i<n; i++){

cout<<a[i]<<" ";

}

}

// Ham hoan doi

void swap(float& x, float& y){

float temp = x;

x=y;

y=temp;

}

// Ham tra ve gia tri int cua float

int round(float f){

return (int)f;

}

// Ham sap noi bot (tang dan)

void sort\_bt(float a[], int n){

for(int i=0; i<n-1; i++){

for(int j=0; j<n-i-1; j++){

if(round(a[j]) < round(a[j+1])){

swap(a[j],a[j+1]);

}

}

}

}

int main(){

float a[100];

int n;

do{

cout<<"Nhap n = ";

cin>>n;

}while(n<0);

input(a,n);

sort\_bt(a,n);

output(a,n);

return 0;

}

Giả sử rằng hàm round(…) là phần tốn nhiều thời gian thực hiện chương trình nhất trong quá trình sắp xếp, tính số lần gọi tới hàm round(…) trong đoạn chương trình trên trong trường hợp xấu nhất và tốt nhất. Sau đó đánh giá độ phức tạp thời gian thực hiện của thuật toán bằng ký hiệu bậc phức tạp O( ). (Chỉ ghi đáp số, không cần giải thích).

T(N) = ???

O(N) = ???