First

Computer-Chocolate Code-Off
of 2023!

Rules:

* Notes allowed.
* Powerpoints allowed
* Googling – okay
* NO RUNNING THE CODE! Figure it out yourself by reading the code!
* You can ask me yes/no questions (and no “is this the answer?”)

The winning team gets 3 pts towards the upcoming Celebration of Knowledge

The 2nd place team gets 2 pts

The 3rd place gets 1 pt

EVERYONE gets chocolate!

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Given the following code, what is the worst case running time?

int func1(int n) {

 int arr[n];

 int sum = 0;

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

 cout << i<< endl;

 arr[i] = n;

 sum += n;

 }

 cout << sum << endl;

 return sum;

}

Ans: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Given the following code, what is the worst case running time?

void func1(int n) {

 int arr[n];

 int sum = 0;

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

 arr[i] = sum;

 sum += n;

 }

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

 cout << arr[i]<< endl;

 }

}

Ans: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. In order to push onto an array, you’d have to make a brand new array that is one longer than the current array, then copy over every single value, and then (finally) add the new value to the end of the array. The worst case running time for this would be:

Ans: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. In order to pop off a doubly linked list, you set a temp pointer to the last node (we can assume we have a last pointer), set last to now point to the temp’s (aka the current last’s) previous node (the one before last), set last’s next to NULL, and either return or delete the temp node.
2. True/False The amount of time this takes is dependent on how many nodes we have in the linked list
3. The worst case running time for this would be: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Given the following, what is printed out? **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

class food{

public:

string fname;

};

void changefood(food x) {

 x.fname = "hot dog";

}

int main() {

food h1;

h1.fname = "nachos";

changefood(h1);

cout << h1.fname << endl; // what is printed here?

return 0;

} //main

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. **Given the following code, what is printed in main?**

void f2(char \*a, char b, char &c);

int main() {

 char x = 's';

 char y = 'u';

 char z = 'l';

 f2(&x,y,z);

 cout << z << y << z << z<<x << endl;

}

void f2(char \*a, char b, char &c) {

 \*a = 'y';

 b = 'o';

 c = 'p';

}

Answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Given the following code:

 **int** size = 5;

 **char** mat[size][size] = { {'t','k','e','w','h'},

 {'o','i','t','n','u'},

 {'n','u','m','w','j'},

 {'p','i','a','n','z'},

 {'t','u','y','r','s'}};

 **for** (**int** i = 0; i < size; i++) {

 **for** (**int** j = 0; j < size; j++) {

 **if** (i == j) {

 cout << mat[i][j] << ",";

 }

 **else** **if** (i == (size-1)-j) {

 cout << mat[i][j] << ",";

 }

 }

 }

 What is printed out? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. The following line of code:

Node \*tmp = new Node(“blug”);

Creates:

|  |  |  |
| --- | --- | --- |
| NULL | “blug” | NULL |

0x3

1. \_\_\_\_\_\_\_\_\_\_\_\_\_True/False: The Node with “blug” as data is located on the stack
2. \_\_\_\_\_\_\_\_\_\_\_\_\_In the above code, new returns:
	1. NULL
	2. “blug”
	3. 0x3
	4. Nothing – it is a command
	5. Hot chocolate, or at least it should

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. ADT stands for:
2. Absurdly Dumb Texts
3. Anxiously Dodging Tapirs
4. Average Denominator Twists
5. Affable Denizens of Teddybear hamsters
6. Abstract Data Types

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. The following is a linked list in memory:



Given the following code:

DNode \*tmp = first;

tmp = tmp->next;

//What address does tmp hold? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

tmp->next->next->prev = tmp;

 //What address was just changed? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 //What address was it changed to? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

tmp->next = tmp->next->next;

 //What address was just changed? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 //What address was it changed to? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 //What did node with data “sou” just become? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

tmp = first;

while (tmp != NULL) {

 cout << tmp->data;

 tmp = tmp->next;

}

//What is printed out? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Given the following code:

**char** arr2[10] = {'t','k','e','w','a','a','i','t','s','x'};

 **for** (**int** i = 0; i < size; i++) {

 dll->push(arr[i]);

 **if** (i%3 != 0) {

 dll->pop();

 }

 }

 dll->printList();

 What is printed out? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. **Given the following linked list:**

u<->c<->r<->o<->p<->e<->n<->i<->p

**And the following code:**

**int** **main**() {

 ... // assume the dll, above, has already been initialized

 dll->afunc2(5);

**}**

**void** **DLL::afunc2**(**int** x) {

 DNode \*tmp = first;

 **for** (**int** i = 0; i < x; i++) {

 tmp = tmp->next;

 }

 last->next = first;

 first->prev = last;

 first = tmp;

 last = tmp->prev;

 first->prev = NULL;

 last->next = NULL;

 /\*\*\* printing \*\*\*\*/

 tmp = last;

 **while** (tmp != NULL) {

 cout << tmp->data<<",";

 tmp = tmp->prev;

 }

 cout << **endl**;

}

 What is printed out? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Given the following:

 The DLL is: e<->c<->h<->i

 n (a second DLL) is: d<->n<->a

Given the following code:

void DLL::unnamedFunc(DLL \*n) {

 last->next = n->first;

 n->first->prev = last;

 last = n->last;

 size += n->size

}

1. If you were to print out the list now, you’d get: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. If you were to print out n->first->data, you’d get: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. A better name than unnamedFunc would be: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
4. True/False: the worst case run time for this function is dependent on the number of nodes in the first and/or the second DLL
5. Thee worst case run time for this function is: \_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Assume you have two linked lists as follows:

list1 is b<->s<->e<->z<->m<->d<->o<->r<->e<->w

list2 is v<->s<->b<->m<->t<->o<->p<->k<->r<->e

And

DLL \*n = new DLL();

DNode \*tmp = list1->first;

DNode \*tmp2 = list2->first;

If you run the following function with:

is(tmp,tmp2,n);

then print out the new list (n), what do you get?

void DLL::is(DNode \*tmp, DNode \*tmp2first, DLL \*n) {

 while (tmp != NULL) {

 tmp2 = tmp2first;

 while (tmp2 != NULL)) {

 if (tmp->data == tmp2->data) {

 if (n->size == 0) {

 n->first = new DNode(tmp->data);

 n->last = n->first;

 }

 else {

 n->last->next = new DNode(tmp->data);

 n->last->next->prev = n->last;

n->last = n->last->next;

 }

 n->size++;

 n->last->next = NULL;

 }

 tmp2 = tmp2->next

 }

 tmp = tmp->next;

 }

 tmp = n->first;

 while (tmp != NULL) { //PRINTS LIST

 cout << tmp->data <<”,”;

 tmp = tmp->next;

 }

}

What is printed out? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Given the following linked list:

t<->m<->i<->w<->a<->a<->u<->t<->f<->n

And the following code:

 first->next->prev->data = last->prev->data;

 DNode \*tmp = first->next;

 last->prev->prev->data = last->data;

 last->data= tmp->data;

 cout <<last->data << ",";

 cout <<last->prev->prev->prev->prev->next->data<<",";

 tmp = last;

 tmp = tmp->prev->prev;

cout << tmp->next->data<<",";

 cout << first->next->prev->data<<",";

 cout << first->next->next->next->prev->data<<",";

cout << tmp->data<<**endl**;

What is printed out? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Given the following code:

 **char** mat[12] = {{'n','u','a'},{'w','a','a'},{'o','g','t'},{'b','u','p'}};

 **for** (**int** i = 0; i < 3; i++) {

 **for** (**int** j = 0; j < 4; j++) {

 dll->push(mat[j][i]);

 }

 }

 DNode \*tmp = first;

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

 cout << mat[tmp->data] << ", ";

 tmp = tmp->next->next;

 }

cout << **endl**;

What is printed out? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_