CSCI 211.
MidtermExam1
75 minutes
Instructor: Dr. Attarwala
California State University, Chico

Enter your First Name: _____

Enter your Last Name: _____

Enter your Student#: _____

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	5	5	5	5	5	30	15	8	10	88
Score:										

- This is a closed book exam
- We will take no questions during the exam. If you have a doubt, state your assumptions and continue writing the exam.

ONLY TURN THIS PAGE WHEN YOU ARE ASKED TO DO SO. You can use this page as an extra sheet if you so require. If you use this page to write your solution, clearly tell us which question this solution is for.

ONLY TURN THIS PAGE WHEN YOU ARE ASKED TO DO SO. You can use this page as an extra sheet if you so require. If you use this page to write your solution, clearly tell us which question this solution is for.

1. Write a C++ function named `swapHalves` that swaps the first half and the second half of an integer array. (5)

Solution:

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

void swapHalves(int *arr, int n)
{
    int *left = arr;
    int *right = arr+(n/2);
    while(left<=(arr+(n/2)-1))
    {
        int temp=*left;
        *left=*right;
        *right=temp;
        left++;
        right++;
    }


    if (right==(arr+(n-1)))
    {

        int rightMostElement=*right;
        while(right>(arr+(n/2)))
        {
            *right=*(right-1);
            right=right-1;
        }
        *right=rightMostElement;
    }
}

int main()
{
    int arr[]={1,2,3,4,5,6,7,8,9};
    swapHalves(arr,9);
   for (int i=0;i<9;i++)
    {
        cout << arr[i] <<endl;
    }

}
```

2. Write a C++ function named `doubleElements` that doubles each element in an integer array. (5)

Solution:

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

void doubleElements(int *arr, int n) {
    for (int i = 0; i < n; ++i) {
        arr[i] *= 2;
    }
}
```

3. Write a function that counts the number of odd numbers in the array. (5)

Solution:

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

int countOdds(int *arr, int n) {
    int count = 0;
    for (int i = 0; i < n; ++i) {
        if (arr[i] % 2 != 0) {
            count++;
        }
    }
    return count;
}
```

4. Write a C++ function named `findMaxMin` that finds the maximum and minimum values in an integer array and prints them. (5)

Solution:

```
#include <iostream>
#include <climits>
using namespace std;

void findMaxMin(int *arr, int n) {
    int maxVal = INT_MIN;
    int minVal = INT_MAX;

    for (int i = 0; i < n; ++i) {
        if (arr[i] > maxVal) {
            maxVal = arr[i];
        }
        if (arr[i] < minVal) {
            minVal = arr[i];
        }
    }

    cout << "Max: " << maxVal << ", Min: " << minVal << endl;
}
```

5. Write a C++ function named `incrementByIndex` that increments each element in an integer array by its index. (5)

Solution:

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

void incrementByIndex(int *arr, int n) {
    for (int i = 0; i < n; ++i) {
        arr[i] += i;
    }
}
```

6. Multiple-Choice Questions

Study the following C++ code from lecture, and answer the following multiple choice questions:

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


int main()
{

    //example 0
    int x=5;
    printf ("The value of x is %i\n",x); //type of x is int
    printf ("The address of x is %p\n",&x); //types involving memory address is
        pointers and pointers store memory address type of &x is int (*)

    //example 1
    int arr1[] ={1,2,3,4}; //type of arr1 is array of ints
    printf ("The address of of arr1 is %p\n",&arr1); //type of &arr1 is int (*)[4]
        i.e., the address of the start of the array
    printf ("The value of of arr1[0] is %i\n",arr1[0]);
    printf ("The value of of arr1[1] is %i\n",arr1[1]);
    printf ("The value of of arr1[2] is %i\n",arr1[2]);
    printf ("The value of of arr1[3] is %i\n",arr1[3]);
    printf ("The address of of arr1[0] is %p\n",&arr1[0]); //type of &arr1[0] is
        int *
    //decay of arr1 into a pointer
    printf ("The value of arr1 is %p\n",arr1); //type of arr1 is int *, and the
        only time this does not happen is when you have a uniary operator such as &
         applied on the name of the array i.e. &arr1
    printf ("The size of is %i bytes\n",sizeof(arr1)); //arr1 does not decay into a
         pointer when used with sizeof
    printf ("The size of is %i bytes\n",sizeof(&arr1));
    printf ("The size of is %i bytes\n",sizeof(&arr1[0]));

    //example 2
    int *ptr = arr1;
    printf ("The address of of ptr is %p\n",&ptr); //type of ptr is int (*)
    printf ("The value of ptr is %p\n",ptr);
    printf ("The size of is %i bytes\n",sizeof(ptr));

}
```

(a) What is the value of `x` in the given code? (3)

1. A) 1
2. B) 5
3. C) 4
4. D) 0

Correct Answer: B) 5

(b) What is the type of `&arr1` in the code? (3)

1. A) int
2. B) int*
3. C) int[4]
4. D) int (*)[4]

Correct Answer: D) int (*)[4]

(c) What is the value of `arr1[3]` in the given code? (3)

1. A) 1
2. B) 3
3. C) 4
4. D) 5

Correct Answer: C) 4

(d) What is the type of `ptr` in the code? (3)

1. A) int
2. B) int*
3. C) int[4]
4. D) int (*)[4]

Correct Answer: B) int*

(e) What will be the output for the `sizeof(&arr1)` in the code? (3)

1. A) 4 bytes
2. B) 8 bytes
3. C) 16 bytes
4. D) It will result in a compilation error

Correct Answer: B) 8 bytes

(f) What is the value of `arr1[0]` in the given code? (3)

1. A) 0
2. B) 1
3. C) 2
4. D) Undefined

Correct Answer: B) 1

(g) What will be the output for `sizeof(arr1)` in the code? (3)

1. A) 4 bytes
2. B) 8 bytes
3. C) 16 bytes
4. D) 32 bytes

Correct Answer: C) 16 bytes

(h) What will be the output for `sizeof(&arr1[0])` in the code? Assume 64 bit architecture. (3)

1. A) 4 bytes
2. B) 8 bytes
3. C) 16 bytes

Correct Answer: B) 8 bytes

(i) What is the value of `ptr` in the given code? (3)

1. A) Address of `arr1[0]`
2. B) Address of `arr1`
3. C) Address of `ptr`
4. D) 1

Correct Answer: A) Address of `arr1[0]`

(j) Which of the following statements is true about array decay in C++? (3)

1. A) The array name always decays into a pointer.
2. B) The array name does not decay when used with `sizeof` or with unary `&`.
3. C) Array decay means the array will be deleted from memory.
4. D) Array decay happens only for character arrays.

Correct Answer: B) The array name does not decay when used with `sizeof` or with unary `&`

7. For each of the following questions, indicate whether there is a memory leak in the given C++ code (answer "Yes" or "No"). If you answer "Yes," provide a solution by writing C++ code that shows how you would fix the memory leak. There is no explanation required if you answer "No".

(a) (5)

```
void bar(int *p)
{
    for (int i = 0; i < 5; i++)
    {
        p[i] = i;
    }
    delete [] p;
}

int main()
{
    //Creates an array of 5 integers
    int *p = new int[5];
    bar(p);
    return 0;
}
```

Solution: No. There is no memory leak. You can confirm this via Valgrind[1]

(b) (5)

```
void bar(int *p)
{
    for (int i = 0; i < 5; i++)
    {
        p[i] = i;
    }
}

int main()
{
    //Creates an array of 5 integers
    int *p = new int[5];
    bar(p);
    delete [] p;
    return 0;
}
```

Solution: No. There is no memory leak.

[1] We look at valgrind in week 2

(c)

```
void bar(int **p)
{
    // Loop 5 times to allocate memory for 5 integer pointers
    for (int i = 0; i < 5; i++)
    {
        // Assign memory address for each integer pointer to an integer
            with value i
        p[i] = new int(i);
    }
}

int main()
{
    // Create an array of 5 integer pointers
    int **p = new int*[5];
    bar(p);

    delete [] p;
    return 0;
}
```

(5)

Solution: Yes. There is memory leak.

8. Multiple Choice Questions

(a) What is the output of the following C++ code? (4)

```
int main()
{
    double arr[5] = {1.0, 2.0, 3.0, 4.0, 5.0};
    double *ptr = arr;
    cout << *(ptr + 1) - *ptr;
}
```

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ None of the above

Correct Answer: A) 1

(b) What is the output of the following C++ code? (4)

```
int main() {
    int x = 1, y = 2;
    int *ptr[2];
    ptr[0] = &x;
    ptr[1] = &y;
    cout << *ptr[0] << " and " << *ptr[1];
    return 0;
}
```

- ◯ 1 and 2
- ◯ 2 and 1
- ◯ address of x and address of y
- ◯ address of y and address of x
- ◯ None of the above

Correct Answer: A) 1 and 2

9. Here is the `LinkedList` class from lecture. Study the code carefully.

   (a) You have been asked to complete the `removeLastNode` function beginning on line #47, which should remove the last node from a linked list [2]. The code for the function has already been partially provided, and you are asked to complete the TODO sections. It is important to ensure that there are no memory leaks in your code. (10)

   If your implementation of `removeLastNode` is correct, then the `toString` function on line #36 will print "4 3 2". Please do not modify any other part of the `LinkedList` class.

```
1  class LinkedList
2  {
3      private: class Node
4      {
5          public:
6              int data;
7              Node *next;
8              Node(int data, Node *next)
9              {
10                 this->data=data;
11                 this->next=next;
12             }
13             ~Node()
14             {
15                 this->next=nullptr;
16             }
17     };
18     Node *head;
19     int size;
20
21     public:
22     LinkedList();
23     addAtHead (int x);
24     string toString();
25     void removeLastNode();
26 };
27 LinkedList::LinkedList()
28 {
29     head=nullptr;
30     size=0;
31 }
32 void LinkedList::addAtHead (int x)
33 {
34     head=new Node (x,head);
35 }
36 string LinkedList::toString()
37 {
38    string ret="";
39    Node *current=head;
40    while(current!=nullptr)
41    {
42       ret=ret+to_string(current->data)+" ";
43       current=current->next;
44    }
45    return ret;
46 }
47
```

[2] This question is identical to what appeared in Lab 5 of this week

```
48 void LinkedList::removeLastNode()
49 {
50    //Linked List is empty, there is nothing to do
51   if (head==nullptr)
52   {
53     return;
54   }
55
56  //Linked List contains exactly one node
57  if (head->next==nullptr)
58  {
59     //TODO. Complete this block (approx 3 lines of code)
60
61
62
63
64
65  }
66  //Linked List contains atleast two nodes
67  else
68  {
69     Node *current=head;
70     Node * previous=nullptr;
71     //TODO. Complete this block (approx 6 lines of code)
72
73
74
75
76
77
78
79
80
81
82
83
84  }
85 }
86
87 int main(void)
88 {
89    LinkedList *l2 = new LinkedList();
90    l2->addAtHead(1);
91    l2->addAtHead(2);
92    l2->addAtHead(3);
93    l2->addAtHead(4);
94    cout <<"Linked List is before calling remove last node: " << l2->toString()
          <<endl;
95    l2->removeLastNode();
96    cout <<"Linked List is : " << l2->toString() <<endl;
97 }
```

Solution

```
void LinkedList::removeLastNode()
{
    // Linked List is empty, there is nothing to do
    if (head == nullptr)
    {
        return;
    }

    // Linked List contains exactly one node
    if (head->next == nullptr)
    {
        delete head; // Delete the only node in the list
        head = nullptr; // Set head pointer to nullptr, as list is now empty
        return;
    }

    // Linked List contains at least two nodes
    else
    {
        Node *current = head;
        Node *previous = nullptr;

        while (current->next != nullptr) // Traverse to the last node
        {
            previous = current; // Keep track of the second last node
            current = current->next; // Move to the next node
        }

        delete current; // Delete the last node
        previous->next = nullptr; // Set the next of the second last node to
            nullptr
    }
}
```