Name: _____

Write all of your responses on these exam pages. If you need extra space please use the backs of the pages.

1 Short Answer: 5 Points Each

1. Write a single line of code that will create an array of 1,000,000 doubles in the heap.

   Solution:

```
int *A = new int[1000000];
```

2. Write a function that takes as parameters an array of integers and the size of the array. Have the function crate a duplicate of the array and pass back a pointer to the new array.

   Solution:

```
int* Copy(int A[], int sz) {
    int *B = new int[sz];
    for (int i = 0; i < sz; i++)
        B[i] = A[i];
    return B;
}
```

3. Write a function that sorts an integer array in ascending order using the selection sort.

   Solution:

```
void selectionSort(int array[], int size) {
    int minIndex, minValue;

    for (int start = 0; start < (size - 1); start++) {
        minIndex = start;
        minValue = array[start];
        for (int index = start + 1; index < size; index++) {
            if (array[index] < minValue) {
                minValue = array[index];
                minIndex = index;
            }
        }
        swap(array[minIndex], array[start]);
    }
}
```

4. Write a function that will take as parameters an array of integers, the size of the array, and a target value to search for. Have the function do a binary search for the target and return the index of the target in the array if found and -1 if not.

   Solution:

```
int binarySearch(const int A[], int size, int value) {
    int first = 0;
    int last = size - 1;
    int middle = 0;
```

```
    while (first <= last) {
        middle = (first + last) / 2;
        if (A[middle] == value) {
            return middle;
        } else if (A[middle] > value)
            last = middle - 1;
        else
            first = middle + 1;
    }
    return -1;
}
```

5. Write a struct that will hold information about an automobile. The data it needs to store is the make, model, year, and cost. Use the appropriate data types for each of the fields. Also give the code that will create an array of 250 cars in the heap.

   Solution:

```
struct car {
    string make;
    string model;
    int year;
    double cost;
};

car *cars = new car[250];
```

6. Write a function that will take in one of the automobile data types you created in the previous exercise by reference but not editable and have the function display the information about the car to the console.

   Solution:

```
void printCarInfo(const car &c) {
    cout << "Make: " << c.make << endl;
    cout << "Model: " << c.model << endl;
    cout << "Year: " << c.year << endl;
    cout << "Cost: $" << c.cost << endl;
}
```

7. Write the declaration of a class structure that will hold information about an automobile. The data it needs to store is the make, model, year, and cost. Use the appropriate data types for each of the fields. Also have the declaration include a default constructor (which stores just 0 for numeric data and the empty string for string data), a non-default constructor that brings in the data for each field as parameters, and a print that prints the car information. Do not implement the functions.

   Solution:

```
#ifndef CAR_H_
#define CAR_H_

#include <iostream>
using namespace std;

class car {
private:
    string make;
    string model;
```

```
    int year;
    double cost;

public:
    car();
    car(string, string, int, double);
    void print();
};

#endif /* CAR_H_ */
```

8. Write the implementations of the two constructors and the print function from the previous exercise. No inline code, this should be written as if the implementation is in a separate cpp file.

   Solution:

```
#include "car.h"

car::car() {
    make = "";
    model = "";
    year = 0;
    cost = 0;
}

car::car(string mk, string md, int y, double c) {
    make = mk;
    model = md;
    year = y;
    cost = c;
}

void car::print() {
    cout << "Make: " << make << endl;
    cout << "Model: " << model << endl;
    cout << "Year: " << year << endl;
    cout << "Cost: $" << cost << endl;
}
```

2 Program Trace: 10 Points

Write the output of the following program.

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

void pt(int A[], int sz) {
    for (int i = 0; i < sz; i++)
        cout << A[i] << " ";
    cout << endl;
}

int main() {
    int A[] {1,2,3,4,5,6,7,8,9};
    int *p1 = A;
    int *p2 = nullptr;
    pt(A, 9);
    cout << *p1 << endl;
    cout << *p1++ << endl;
    cout << *p1++ << endl;
    cout << *++p1 << endl;
    cout << ++(*p1) << endl;
    pt(A, 9);
    p1 = A;
    p2 = p1 + 6;
    while (p1 < p2){
        (*p2) += 2;
        p2 -= 2;
    }
    pt(p1, 9);
    p1 = A;
    for (int i = 0; i < 9; i++)
        *p1++ = i;
    pt(p2, 5);

    return 0;
}
```

Output

```
1 2 3 4 5 6 7 8 9
1
1
2
4
5
1 2 3 5 5 6 7 8 9
1 2 5 5 7 6 9 8 9
0 1 2 3 4
```

3 Programming: 50 Points

Create a struct called `IntList` that stores an integer size and a pointer called `list` to an integer (array). The size is the amount of space in the array that is pointed to by list. As a default, the size is to be set to 0 and the pointer to nullptr. Create the following functions that would accompany this structure.

- `get`: Will take an IntList and a position and return the element in the array at that position. If the position is out of bounds the function should display an error that you are out of bounds and return 0.
- `set`: Will take an IntList, a position, and a value. It will put the value in the given position of the list array if the position is within the bounds of the array and if the position is not the array will not be altered. This function is to also print an out of bounds error in this case.
- `get_size`: Will take an IntList and return the size of the list.
- `concat`: Will take in two IntList variables, concatenate the second list onto the first and not alter the second list.
- `resize`: Will take an IntList and a new size for the list. The function will reset the size of the array that is pointed to by the list field to the new size that is given. If the new size is smaller than the old the new list will contain the same data as the old but be truncated to the new size. If the new size is larger than the old the new list will be the old list padded with zeros on the end. If the new size is 0 or negative the function will print out an error and no alteration to the list will be made.
- `sort`: Will take an IntList and sort the list of integers in ascending order.
- `find`: Will take an IntList and target value to be searched for. The function will return the index of the element if found and -1 if the target value is not in the list.
- `destory`: Will take an IntList and release the memory that was allocated to it.
- `print`: This will print the list on a single line with a space or two between the elements.

As an example the following program will produce the following output.

```
int main() {
    IntList nums;
    int size = 7;

    resize(nums, size);
    for (int i = 0; i < size; i++) {
        set(nums, i, i + 1);
    }
    print(nums);
    resize(nums, 5);
    print(nums);
    resize(nums, 8);
    print(nums);
    set(nums, 10, 20);
    print(nums);
    set(nums, 4, 20);
    set(nums, 2, 7);
    set(nums, 7, 1);
    print(nums);
    sort(nums);
    print(nums);
    cout << find(nums, 4) << endl;
    cout << find(nums, 5) << endl;
    cout << get(nums, 2) << endl;
    cout << get(nums, 7) << endl;
    cout << get(nums, 20) << endl;

    IntList nums2;
    resize(nums2, 4);
    for (int i = 0; i < 4; i++) {
        set(nums2, i, 20 - i);
    }
    print(nums);
    print(nums2);
    concat(nums, nums2);
    print(nums);
    print(nums2);

    destroy(nums);
    destroy(nums2);
    return 0;
}
```

Output

```
1 2 3 4 5 6 7
1 2 3 4 5
1 2 3 4 5 0 0 0
List bounds error.
1 2 3 4 5 0 0 0
1 2 7 4 20 0 0 1
0 0 1 1 2 4 7 20
5
-1
1
20
List bounds error, returning 0.
0
0 0 1 1 2 4 7 20
20 19 18 17
0 0 1 1 2 4 7 20 20 19 18 17
20 19 18 17
```

Solution:

```
#include <iostream>
#include <algorithm>

using namespace std;

struct IntList {
    int size = 0;
    int *list = nullptr;
};

void set(IntList &iList, int pos, int item) {
    if (pos >= iList.size || pos < 0) {
        cout << "List bounds error." << endl;
    } else {
        iList.list[pos] = item;
    }
}

int get(const IntList &iList, int pos) {
    if (pos >= iList.size || pos < 0) {
        cout << "List bounds error, returning 0." << endl;
        return 0;
    } else {
        return iList.list[pos];
    }
}

int get_size(const IntList &iList) {
    return iList.size;
}

void concat(IntList &iList1, const IntList &iList2) {
    int *iPtr = iList1.list;
    int newcap = iList1.size + iList2.size;
    iList1.list = new int[newcap];
    int j = 0;
    for (int i = 0; i < iList1.size; i++)
        iList1.list[j++] = iPtr[i];

    for (int i = 0; i < iList2.size; i++)
        iList1.list[j++] = iList2.list[i];

    iList1.size = newcap;
    delete[] iPtr;
}

void resize(IntList &iList, int newsize) {
    if (newsize <= 0) {
        cout << "Cannot resize to length 0 or less." << endl;
        return;
    }

    int *iPtr = iList.list;
    iList.list = new int[newsize];
    int min = (iList.size < newsize) ? iList.size : newsize;

    for (int i = 0; i < newsize; i++)
        iList.list[i] = 0;

    for (int i = 0; i < min; i++)
        iList.list[i] = iPtr[i];

    iList.size = newsize;
```

```
64
65      delete[] iPtr;
66  }
67
68  void destroy(IntList &iList) {
69      delete[] iList.list;
70      iList.list = nullptr;
71      iList.size = 0;
72  }
73
74  void sort(IntList &inlist) {
75      sort(inlist.list, inlist.list + inlist.size);
76  }
77
78  int find(const IntList &inlist, int target) {
79      for (int i = 0; i < inlist.size; i++)
80          if (inlist.list[i] == target)
81              return i;
82      return -1;
83  }
84
85  void print(IntList &iList) {
86      for (int i = 0; i < iList.size; i++)
87          cout << iList.list[i] << " ";
88
89      cout << endl;
90  }
```