CS 240 Data Structure

Spring 2020 Exam I

03/23/2020

This exam contains **three sections**

A) Code: (basic data type, pointer, ADT)
   a. Reading: Trace the code to predict the output of the code
   b. Filling: Fill in the missing code to guarantee the shown output
   c. Quick short answer telling the difference of the implementation
   d. If the program won’t compile, simply say so.

B) Algorithm: (critical thinking and analysis ability)
   a. You are given a sorting problem and you are supposed to design strategies to solve this problem
   b. Briefly talk about how efficient it is. For instance, if the major computation is 2 for loops and each loop contains N iterations, then the complexity is approximately N^2.
   c. You are provided enough space to write out your solution.

C) Concepts

⇨ Good luck and do your best.

⇨ When you are lost in pointers and addresses, the best way to attack that is simply drawing the diagram with respect to address of the variable that the pointer points to, and the pointer’s its own address and the dereferencing operator *.

Name:

Score: /105 + 5

A) Basic Data Type, Pointers and ADT: Problem 1: 10pts (5+5)

```
#include <iostream>

using namespace std;

void changethis(int* p)

{ for(int i = 0; i < 15; i++)

   {  What to fill in here? Answer 1:_______________________________________________(5pts)

p++;   }

}

int main ()

{

  int numbers[5];   int * p;

  p = numbers;  *p = 10;  p++;  *p = 20;  p = &numbers[2];  *p = 30;  p = numbers + 3;  *p = 40;

  p = numbers;  *(p+4) = 50; p++ ;

cout<<*(p+4)<<endl;  // What will be printed out here ?? Answer 2: ___________________________(5pts)

 int s[15];   int *t;   t = s;

  changethis(t);

   for(int i = 0; i < 15; i++)

    { cout<<s[i]<<endl; } //Here we prints out 2, 4, 6,, ..., 30, What should you do in Answer 1

  return 0;

}
```

A) Basic Data Type, Pointers and ADT: Problem 2: 10 points

```
#include <iostream>

#include <string>

using namespace std;

void prntarray(int *k, int size);

int main()

{    int thesize = 10;

    int myarray[thesize];

    int *p = myarray;

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

    {       myarray[i] = i;     }

prntarray(p, thesize);

 return 0;

}
```

Please write out the function prnarray that will print out the elements inside the myarray array

```
Void prnarray(int* p, int t)

{

Line 1:

Line 2:

Line 3:

Line 4:

Line 5:

Line 6:

Line 7:

}
```

A) Basic Data Type, Pointers and ADT: Problem 3: Pass by reference, pointer and variables (15pts)

```
// passing parameters by reference
#include <iostream>
using namespace std;
void duplicate (int& a, int& b, int& c)
{  a=2;  b=2;  c=2;}
void duplicate1 (int a, int b, int c)
{  a=5;  b=6;  c=7;}
void duplicate2 (int* a, int* b, int* c)
{  *a = 3;   *b = 3;   *c = 3; }
int main ()
{  int x=1, y=3, z=7;  int *d, *e, *f;
  d=&x;   e=&y;   f=&z;
  duplicate (x, y, z);
// Please predict the output
  cout << "x=" << x << ", y=" << y << ", z=" << z << endl;
```
Answer 3: ______________________________ 5pts
```
  duplicate1 (x, y, z);
  cout << "x=" << x << ", y=" << y << ", z=" << z<<endl;
```
Answer 4: ______________________________ 5pts
```
  duplicate2 (d,e,f );
  cout << "x=" << x << ", y=" << y << ", z=" << z<<endl;
```
Answer 5: ______________________________ 5pts
```
  return 0;
}
```

A) Basic Data Type, Pointers and ADT: Problem 4: Address of an array, &s means the address of s

```
int myarray[thesize];
int *p = myarray;
```

Answer 6: True / False: &p = &myarray[0] and myarray = &myarray[0] **(5pts)**

B) Algorithm: Problem 1: Sorting (40 pts + 5 bonus pts)

In PS1, we talked about LinkedList and we tried to sort a given array of integer. Suppose we are given an array as the following:

int test[] ={4, 8, 2, 1, 5, 6, 66, -11, 34, 7, 3, 21, 17, 7};

Please write the pseudocode to create a linklist that after the insertion the head is 4 and tail is 7.

Answer 6: (21pts)

```
#include <iostream>
using namespace std;
struct Node {
   int data;
   Node *next;
};

Node* head = NULL;
Node* tail = NULL;

void insert(int new_data) {
    Line 1: (mem allocation)
    Line 2:(data assignment)
    Line 3:(next point assignment)
    if(head == NULL)
    {
      Line 4: (head assignment)
      Line 5: (tail assignment)
    }

    else
    {
      Line 6: (tail next assignment)
      Line 7: (tail assignment)
    }

}

void display() {
   Node* ptr;
   ptr = head;
   while (ptr != NULL) {
      cout<< ptr->data <<" ";
      ptr = ptr->next;
   }
}
```

In PS1, we talked about LinkedList and we tried to sort a given array of integer. Please modify the insert function accordingly.

Answers: (10points)

```
void insert(int new_data) {

}
```

What is the complexity of your sorting solution and why?

Bonus Answer 6: (5 pts)

Memory management: Let say we you want to remove the value 66 from the list. Please show how you would access it and then free the memory that item “66” occupies (maybe you can traverse using the display function as well)? [pure description: ½ credits; correct pseudo code: full credits]

Answer 7: (9 pts: 4 + 5)

C) Class: Queue and Stack and List : 20pts

Suppose have a stack object st and a queue object q. We perform the following operations.

```
For (int j = 1; j <=11, j++}
{ st.push(j);
q.enqueue(2*j);
}
q.enqueue(st.top());
st.pop();  → output 1
st.push(q.dequeue());
q.enqueue(231);
 st.pop(); → output 2
q.dequeueu(); → output 3
```

What are the values of those outputs?

Answer:

Output 1:

Output 2:

Output 3:

D) Postfix

One of the application of stack is postfix that helps building an unambiguous tree for the compiler. Let say A=10, B=5, C=3, D =1. Please compute the following expression:

A B C + * =

A B C D --+ =

C) Dynamic allocation for constructor: 10pts

```
#include <iostream>
using namespace std;
typedef int ElementType;
const int UpperBound = 101;

class List
{
 public:
   List(int maxSize = UpperBound);
   ~List();
   List(const List & origList);
   bool empty() const;
   void insert(ElementType item, int pos);
   void erase(int pos);
   void display() const;

 private:
   int mySize;                // current size of list stored in array
   int myCapacity;            // capacity of array
   ElementType * myArrayPtr;  // pointer to dynamically-allocated array

};

//--- Definition of class constructor
List::List(int maxSize)
: mySize(0), myCapacity(maxSize)
{
   myArrayPtr = new(nothrow) ElementType[maxSize];
}
```

assuming the rest of the implementation of functions for this class is as usual. Suppose you do the following in the main function

List alist();

→ Q1: What is the value of myCapacity?

List blist(123);

→ Q1: What is the value of myCapacity?