Name/Surname: Id:

CENG 112 – DATA STRUCTURES

SPRING 2016-2017 / Midterm Exam Solutions

07.04.2017

- Exam duration is 100 minutes
- No written notes
- No electronic devices
- ...Good Luck...

Q1. (20 Points) Arrays and Pointers

Fill in the empty spaces in the comments below with the values of the expressions at that point in the program.

```
int main() {
  int a = 5;
  int b = 8;
  int c[3] = { 1, 2, 3 };
  int *p1 = &a;
  int *p2 = p1;
  int *p3 = &c[1];

  // *p1 = …5……, *p2 = …5……, p3[0] = …2……

  p3 = p3 – 1;
  a += 3;
  for (int i = 0; i < 2; ++i)
    c[i] = c[i+1]+1;

  // a = ……8…, b = …8……, *p3 = …3……, p3[1] = …4……, p2[0] = …8……, c[0] = …3……

  p1 = &a;
  p2 = &b;
  p3 = p1;
  int *d[3] = { p1, p2, p3 };
  *d[2] += 1;

  // a = ……9…, b = …8……, c[0] = …3……, c[1] = …4……, c[2] = …3……

  return 0;
}
```

Q2. (20 Points) Abstract Data Types

Implement the template class Polynomial with the following public api

```
Polynomial<T>::Polynomial(int degree, const string& name);
void Polynomial<T>::set_coefficients(const T* coeff);
T    Polynomial<T>::evaluate(T x);
void Polynomial<T>::print() const; // prints the polynomial
```

The following program should compile and produce the output

```
p(x) = 2.0*x^2 + 1.0
p(3) is 19
```

```
int main() {
    double p_coeff[3] = { 1.0, 0.0, 2.0 }; // 2x^2 + 1

    Polynomial<double> p(3, “p”);
    p.set_coeff(&p_coeff[0]);

    p.print();
    cout << “p(3) is ” << p.evaluate(3.0) << endl;

    return 0;
}
```

You may use std::vector in your implementation, the number of significant digits in your output does not matter.

Answer:

```
template <typename T>
class Polynomial {
public:
        Polynomial(int degree, const string& name);
        void set_coeff(const T* coeff);
        T    evaluate(T x);
        void print() const; // prints the polynomial
private:
        int m_degree;
        string m_name;
        vector<T> m_coeff;
};

template <typename T>
Polynomial<T>::Polynomial(int degree, const string& name)
{
        m_degree = degree;
        m_name = name;
        m_coeff.resize(degree);
}

template <typename T>
void Polynomial<T>::set_coeff(const T* coeff)
{
        for (int i = 0; i < m_degree; ++i)
                m_coeff[i] = coeff[i];
}


template <typename T>
T Polynomial<T>::evaluate(T x)
{
        T r = 0.0;
        T term = 1.0;
        for (int i = 0; i < m_degree; ++i) {
                r += m_coeff[i] * term;
                term *= x;
        }
        return r;
}

template <typename T>
void Polynomial<T>::print() const
{
        cout << m_name << "(x) = ";
        for (int i = m_degree-1; i >= 0; --i) {
                if (m_coeff[i] != 0.0) {
                        if (i > 1)
                                cout << m_coeff[i] << "*x^" << i << " + ";
                        else if (i == 1)
                                cout << m_coeff[i] << "*x + ";
                        else
                                cout << m_coeff[i] << " ";
                }
        }
        cout << endl;
}
```

Q3. (20 Points) Functions and Dynamic Memory

Fill in the functions f1(), f2(), and f3() according to the comments.

```
// allocate a double-precision floating point number array of length n, fill it
with the numbers (n-1), (n-2), ..., 0 and return the array.
double *f1(int n) {

      double *x = new double[n];
      for (int i = 0; i < n; ++i)
            x[i] = n – 1 – i;
      return x;

}

// Sum the corresponding elements of the arrays ‘a’ with na elements and ‘b’ with
// nb elements and store the result in the array ‘a’. You should stop at the end
// of the shorter array and return the number of summed items. Running on arrays
// a = { 1,2,3 } and b = {4, 5} should produce a = { 5,7,3 } and return 2
int f2(int na, double *a, int nb, const double* b) {

      int n = na;
      if (nb < n)
            n = nb;
      for (int i = 0; i < n; ++i)
            a[i] += b[i];
      return n;

}


// Sum the corresponding elements of the arrays ‘a’ with na elements and ‘b’ with
// nb elements and store the result in a newly allocated array ‘c’. You should
// stop summing at the end of the shorter array and return the array ‘c’.
// a = { 1,2,3 } and b = {4, 5} should produce and return c = { 5,7,3 }
// You can call f2() to implement f3() but it is not mandatory.
double *f3(int na, const double *a, int nb, const double * b) {
      double *c = 0;
      if (na > nb) {
            c = new double[na];
            for (int i = 0; i < na; ++i)
                  c[i] = a[i];
            f2(na, c, nb, b);
      } else {
            c = new double[nb];
            for (int i = 0; i < nb; ++i)
                  c[i] = b[i];
            f2(nb, c, na, a);
      }
      return c;

}
```

Name/Surname: Id:

Q4. (20 Points) Stacks and Queues

a) Write the output of the following program that uses ceng112::Stack and ceng112::Queue.
b) What would be the output if you replaced Queue q with a Stack?

```
void print_if_nice(const string& str)
{
      Stack<char> s;
      Queue<char> q;

        int n = str.size();
        int left_end = n/2;
        int right_start = (n+1)/2;
        for (int i = 0; i < left_end; ++i)
                s.push(str[i]);
        for (int i = right_start; i < n; ++i)
                q.enqueue(str[i]);

        bool is_nice = true;
        for (int i = 0; i < left_end; ++i) {
                if (s.pop() != q.dequeue()) {
                        is_nice = false;
                        break;
                }
        }

        if (is_nice)
                cout << "'" << str << "' is nice" << endl;
        else
                cout << "'" << str << "' is NOT nice" << endl;
}

int main() {
        print_if_nice("abababab");
        print_if_nice("aabbabbaa");
        print_if_nice("abaaba");
        print_if_nice("XZZYXXYZZX");
        print_if_nice("XYZZXXYZZX");
        print_if_nice("");
        print_if_nice(" ");

        return EXIT_SUCCESS;
}
```

Answer:

```
'abababab' is NOT nice
'aabbabbaa' is nice
'abaaba' is nice
'XZZYXXYZZX' is nice
'XYZZXXYZZX' is NOT nice
'' is nice
' ' is nice
----------------------
'abababab' is nice2
'aabbabbaa' is NOT nice2
'abaaba' is nice2
'XZZYXXYZZX' is NOT nice2
'XYZZXXYZZX' is nice2
'' is nice2
' ' is nice2
```

Q5. (20 Points) Linked Lists and Recursion

```
struct Node {
      unsigned int data;
      Node *rest;
};
```

a) Implement print_list_reverse() **RECURSIVELY** so that it prints elements in reverse order.

```
void print_list_reverse(const Node *n) {

        if (head == 0)
                return;

        print_list_reverse(head->rest);
        cout << head->data << " ";

}
```

b) Implement max_element **RECURSIVELY** to calculate&return the maximum of list data.

```
unsigned int max_element(const Node *n) {

        if (head == 0)
                return 0;

        unsigned int max_rest = max_element(head->rest);
        if (max_rest > head->data)
                return max_rest;
        else
                return head->data;

}
```

c) Implement list_free to **ITERATIVELY** free allocated memory.

```
void free_list(Node *n) {

        while (head) {
                Node *p = head;
                head = head->rest;
                delete p;
        }

}
```