Data Structures - Midterm

Do questions 1.1-1.6

1.1 Assume the declaration

```
int *arr;
```

Which declaration dynamically creates an array of 35 integers? Circle one.

(a) `arr = new int[35];`

(b) `*arr = int[35];`

(c) `arr = new int(35);`

(d) `*arr = new[35] of int;`

1.2 Identify the error that occurs in each of the following statements.

(a)
```
vector<int> *vPtr;
vPtr = new vector<int>[15];
```

(b)
```
int a[20], b[10];
b = a;
```

(c)
```
int *p = new int{1,4,9};
```

1.3 Consider the function `f()` for the list object `alist`.

```
void f(list<int>& alist)
{      list<int>::iterator iter ;
       int j;

       while(alist.size() != 0)
       {
              iter = alist.begin();
              for(j = 0 ;  j < alist.size()-1; j++)
                     iter++;
              cout << *iter << "  ";
              alist.erase(iter);
       }
}
```

Assume list `alist` has values `7 12 9 8 15`. What is the output from `f()`?

1.4 A stack is used to reorder an array of integers. Describe the ordering in the integer array `arr` after executing function `f()`?

```
void f(int arr[], int n)
{       stack<int> stk;
        int i;

        for (i = 0; i < n; i++)
                stk.push(arr[i]);

        i = 0;
        while (!stk.empty())
        {       arr[i] = stk.top();
                stk.pop();
                i++;
        }
}
```

(a) creates an ordered array in ascending order.

(b) creates an ordered array in descending order.

(c) reverses the order of the elements.

(d) maintains the same order.

1.5 Consider the following function `f`.

```
template <typename T>
void f(vector<T>& v) {
  int i, n, temp = v[0];
  n = v.size();
  for (i = 0; i < n-1; i++){
    v[i] = v[i+1];
  }
  v[n-1] = temp;
}
```

Assume the vector `v` has values <`10, 20, 30, 40`>. What is the content of `v` after the function call `f(v)`?

1.6 Write the infix expression `a*(b-c)^d` in postfix form.

Question 2

Consider the following `accumulator` class.

```
template <typename T>
class accumulator {
public:
  accumulator(const T& value = T());  // constructor
  T getTotal() const;                 // return total
  void addValue(const T& value);      // add value to total
private:
  T total;
};

template <typename T>
accumulator<T>::accumulator(const T& value): total(value)
{}

template <typename T>
T accumulator<T>::getTotal() const {
  return total;
}

template <typename T>
void accumulator<T>::addValue(const T& value){
  total += value;
}
```

Write one program that does the following: Declares a vector of 10 accumulator objects named `count`. Generate a million random integers in the range from 0 to 9. For each `value`, increment the accumulator `count[value]`. Output `total` of each accumulator in the vector. You can use the `rand()` function in `<stdlib.h>` and the expression `rand()%10` to generate integers.

Question 3

Write a function named `max` that returns the maximum of the data values in the given list `alist`.

```
template <typename T> T max(const list<T>& alist);
```

Question 4

A compiler can use a stack to check whether an expression correctly matches the left and right parentheses. For instance:

```
(a + (b * c)) // Matching parentheses
(a + (b * c) + d // Missing right parenthesis
((a + (b * c)) // Missing left parenthesis
```

Write a function, named `testMatch`, that takes an expression as a string and outputs a message indicating whether the parentheses match or whether the expression is missing a left or right parentheses.