ENSF 337 – Programming Fundamentals for Software and Computer

Samples of Previous Years Final Exam Questions

Note: this document contains some sample questions from previous years, and it aims to serve as additional exercises to get prepared for final exam by trying to solve them. The best way to improve your problem solving skill is to try solving the problems not looking at the solutions and memorize them. Solutions are not available.

IMPORTANT NOTES:

For all of questions in this exam you can make the following assumptions:

- Any needed library header files are included, followed by: `using namespace std;`
- Size of `int` type is 4 bytes, size of `pointer` is 8 bytes, and size of `double` is 8 bytes.
- ASCII value of A is 65, a is 97, and 0 (zero) is 48

SECTION I – Multiple Choice – (1 mark each – total 16 marks)

1. What is the output of the following code segment:

```
const char* s[] = {"NBY", "SDT"};
const char** z = s;
cout << z[1][1];
cout <<    *(z[1]+ 2);
cout <<    (*z + 1) ;
```

a. DSDT
b. DTY
c. DTBY
d. DTS
e. None of the above

2. What is the output of the following program:

```
void fun(int n){
    if (n <= 0) return;
    cout << n + 1;
    fun (n-3);
    cout << n-1;
}
```

```
int main()
{
     fun (5);
     return 0;
}
```

a. 6304
b. 6314
c. 6341
d. 6340
e. 6214
f. None of the above

3. Consider the statement: `char x[] = {'A','B', 'C', '\0', 'D', '\0'}, y[] = "ABC";`
Which of the following is true in C/C++? Assume: `strcmp` returns a positive number if x is greater than y, and a negative number if x is less than y, otherwise it returns zero.

a. `x != y`
b. `strcmp(x, y) == 0`
c. `sizeof(y) < sizeof(x)`
d. `strlen(y+1) < strlen(x)`
e. All of the above is true
f. None of the above is true

4. Which one of the following statements is true:
An assignment operator in a C++ class must:

a. return a reference to `*this` and must avoid self-copying
b. return a reference to `this` and must avoid self-copying
c. return a reference to `this` and must do self-copying
d. return a reference to `*this` and must do self-copying
e. None of the above are correct

5. Consider the following code segment and select the best answer:

```
string s1 = "863";
```

```
s1 += "79";
int y = s1.at(4) - s1.at(2);
```

a. This code gives a compilation error on the second line.
b. The value of y after this code segment will be 6
c. The value of y after this code segment will be -6
d. The value of y after this code segment will be 1
e. None of the above

Use the following code segment to answer questions 6-9 (select the BEST answer):

```
1  char a[10] = "March";
2  const char* s =  a;
3  char *const p = a;
4  *(s-1) = 'U';
5  *p = 'M';
6  p = s;
```

6. Which one of the following statements is true for line 3:
   a. This line gives compilation error.
   b. This line gives runtime error.
   c. None of the above is true.

7. Which one of the following statements is true for line 4:
   a. This line gives compilation error.
   b. This line gives runtime error.
   c. None of the above is true.

8. Which one of the following statements is true for line 5:
   a. This line gives compilation error.
   b. This line gives runtime error.
   c. None of the above is true.

9. Which one of the following statements is true for line 6:
   a. This line gives compilation error.
   b. This line gives runtime error.
   c. None of the above is true.

10. Consider the following code segment:

```
const char *s = "ON TIME";
const char** m = &s;
```

Which one of the following C++ code, output character `'N'` on the screen in a `cout` statement:
   a. `(*(*m+1)+1)`
   b. `*(*m+1)`
   c. `**m + 1`
   d. `m[0][1]`
   e. b and d
   f. c and d
   g. None of the above

11. What will be the output of this program?

```
void fun(int n) {
    if (n <= 0) return;
    cout << "1";
    fun (n-2);
    cout << "2";
}
```

```
int main()
{
    fun (3);
    return 0;
}
```

a. 1112
b. 1212
c. 1122
d. None of the above.

Use the following code segment to answer questions 12 to 13 (select the BEST answer)

```
1  void main(void){
2      double a = 11, b = 22, c = 33, d =44;
3      double *x [4] = {&a, &b, &c, &d};
4      double** y = x +1;
5      cout << y[1][0] << endl;
```

```
6     cout << **(y+2) << endl;
7  }
```

12. Which one of the following statements is correct?
    a. output in line 5 is: 22
    b. Program output in line 5 is: 33
    c. Program output in line 5 is: 11
    d. None of the above statements is correct.

13. Which one of the following statements is correct?

    a. Program output at line 6 is: 44
    b. Program output at line 6 is: garbage
    c. Program output at line 6 is: 11
    d. None of the above statements is correct

Consider the following code segment and answer questions 14 and 15.

```
class Point {
public:
  Point(double x = -99, double y = -99) { this -> xM = x; this -> yM = y;}
  double getx() {return xM;}
  double gety() {return yM;}
  void setx(double x) {this -> xM = x;}
  void sety(double y) {this -> yM = y;}
private:
  double xM, yM;
};
```

14. What is the output of the following code snippet:

```
Point p1(100);
cout << p1.getx() << "  " << p1.gety();
```

    a. The output of the program is: -99 -99
    b. The output of the program is: 100 100
    c. The output of the program is: -99 100
    d. The output of the program is: 100 -99
    e. None of the above. It doesn't compile because constructor of Point needs two arguments.

15. How many times does the constructor of class Point get called by the following code snippet:

```
Point a(100, 200);
Point b[6];
Point *c = new Point;
Point *d = new Point(300, 400);
```

    a. Four times
    b. Three times
    c. Nine times
    d. Two times
    e. None of the above

16. What is output of the following code segment:

```
typedef vector<int> ROW;
vector<ROW> x(3, ROW(5)); // creating a vector of 3 ROW with 5 columns

// populating the matrix with integer numbers
for(int i = 1; i <= 3; i++)
   for (int j = 1; j <= 5; j++)
       x.at(i-1).at(j-1) = i+j;

for(int i = 0; i < 3; i++)
    for (int j = 0; j < 5; j++)
       if(i == j)
          cout << x[i].at(j);
```

    a. 246
    b. 035
    c. 136
    d. None of the above

SECTION II – Short Answer Questions – 8 marks

Consider the following code segment in **C++**, and answer the following question assuming that file `data.bin` has been successfully opened:

```
int a[] = {199, 299, 399, 499, 500, 699, 799, 899, 999};
int* x[3] = {a, a+2, a+5};
ofstream out ("data.bin", ios::binary); // open binary file
out.write((char*)x[2], sizeof(int));
out.close();
```

Question 1 (1 mark): What does this code write into the file `data.bin`?

33 39 39 (???) Getting 73 when testing code...

Question 2 (3 marks). Consider the following C program:

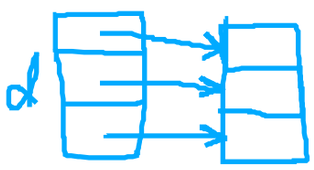
```
// copystr is supposed to return a pointer to the copy of source string

char * copystr (const char* source) {
   char *dest;
   strcpy(dest, source);
   return dest;
}

int main(void) {
   const char *s1 = "ABCD";
   char *s2 = copystr(s1);
   // more code, as needed ...
   return 0;
}
```

The function `copystr` is defective; calling it will most likely result in a runtime error. In the following space, write a code fragment that can be inserted into `copystr` to fix the defect. You may call whatever C library functions you find useful.

dest=malloc(sizeof(char));

Question 3 (2 marks): Consider the following C++ code fragment.

```
int *d[3];
for(int j = 0; j < 3; j++) {
    d[j] = new int;
    **(d+j) = sizeof(d[j]) + j;
    cout <<  *d[j] << endl;;
}
```



Assuming that all the operations of `new` succeed, what is the output?

8910

Question 5 (2 marks): Consider following C++ code and based on given function interface comment write the implementation of the constructor for class Node:

```
class List {
 class Node{
  public:
    Node(int itemA, Node* next );
   //PROMISES: allocates memory for a
   //node, copies value of itemA into
   // itemM and sets nextM to null;
  private:
         Node* nextM;
         int dataM;
     }; // end of class Node
    Node* headM;
public:
     List(): headM(nullptr) {}
     // Assume more functions go here
};// end of class List
```

// Write your answer in this box

Node (int itemA, Node* next){ Node newNode=new N

SECTION III (total of 12 marks)

Part 1 (7 marks). Write a definition for the following C++ function:

```
bool up_then_down(const int* arr, int n);
// REQUIRES: n >= 1; elements arr[0] ... a[n-1] exist.
// PROMISES: Returns true if the sequence of element values is strictly increasing
// from a[0] to the first appearance of the maximum value, then strictly decreasing
// from the first appearance of the maximum value to a[n - 1].
// Otherwise, returns false. EXAMPLES (maximum values are bold):
// The return value would be true for all of these sequences ...
// {10}, {10, 20}, {20, 10}, {10, 20, 30, 25}, {10, 20, 30, 25, -2}
// But would be false for all of these ...
// {10, 20, 10, 15}, {10, 10, 20, 15}, {10, 20, 20, 15}.
```

```
bool up_then_down(const int* arr, int n){     for(int i=0; i<n; i++){          if(arr[i]>arr[i+1]){               return fals
```

Part 2 (5 marks). Write a definition for the following C++ function. In this part, **you may not make any calls to library functions**.

```
bool all_diff(const char *left, const char *right);
// REQUIRES: left and right each point to the beginnings of C strings.
// PROMISES: Return value is true if none of the characters in the left string
// also appear in the right string. ('\0' characters are not included in the
// comparison.)
// If there is at least one match, return value is false.
```

```
bool all_diff(const char *left, const char *right){     for(int i=0; left, right; i++){          if(left[i] == right[i]){
```

SECTION IV (total of 15 marks)

Below is a complete `.h` file for a class called `IntVector`, that stands for Integer Vector, along with a `.cpp` file that has some but not all of the member function definitions needed for the class:

```
// File: intVector.h
class IntVector {
public:
    IntVector() : storeM(0), end_storeM(0) { }

    IntVector(const int *begin, const int *end);
    // REQUIRES: begin and end point to the elements of the same array, in a way
    // that: end – begin equals the number of elements of the array.
    // PROMISES: storeM points to a space allocated on the heap, where elements 0 to
    // size()-1 of this space are initialized using values of *begin, *(begin+1),
    // ..., *(end-1)

    IntVector(const IntVector& src);

    IntVector& operator=(const IntVector& rhs);

    ~IntVector();

    int size() const { return (int) (end_storeM - storeM); }

    const int& at(int i) const { return storeM[i]; }

    int& at(int i) { return storeM[i]; }

    void push_back(int el_val);
    // PROMISES: Size of vector is increased by one element.
    // Last element of vector is equal to el_val.

    void remove_all(int val);
    // PROMISES: If one or more elements match val, all of those elements are
    // removed from the vector (allocated memory on the heap will be reduced).If no
    // elements match val, there is no change to the vector.

private:
    int *storeM;
    int *end_storeM;
};
```

```
// File: intVector.cpp
#include "intVector.h"
IntVector::IntVector(const int *begin, const int *end)
: storeM(0), end_storeM(0)
{
    if (begin == end)
        return;
    size_t count = end - begin;
    storeM = new int[count];
    end_storeM = storeM + count;
    for (size_t i = 0; i < count; i++)
        storeM[i] = begin[i];
}
```

```
int main(){
    int a[] = {12, 25, 18, 25, 30, 2};
    IntVector v2(a, a + 6);
    for(int i =0; i < v2.size(); i++)
        cout << v2.at(i) << endl;
    return 0;
}
```

Program output is:

```
12
25
18
25
30
```

Please, first read the given code and notice the program output, then answer the questions on the next page . . .

Note: Marks will be deducted for redundant, and inefficient code

Question 1 (5 marks). Write a definition for the member function **`push_back`**.

```
int *temp=storeM;storeM++;storeM=new int;while(temp){     *storeM=*temp     storeM++;     temp++;}
```

Question 2 (5 marks). Write the definition of an **`assignment operator=`** for class IntVector.

```
if(this !=rhs){     delete storeM;     delete end_storeM;     storeM=0;     end_storeM=0;     while(rhs->store
```

Question 3 (5 marks). Write a definition for the **remove_all** function of IntVector.

```
int *temp=storeM;while(temp){     if(temp==val){          temp=0;     }}for(int i=0; temp; i++){     if(temp==0){
```

Section V - Linked List (20 marks)

Consider the **partial** definition of a class called `ExamList` and a definition of class `Node`, then answer the following questions:

```
class Node {
public:
  Node(int aItem): itemM(aItem), nextM(0) {}
  int itemM;
  Node *nextM;
};

class ExamList {
public:
 ExamList(): headM(0) {};
~ExamList();
 ExamList& operator= (const ExamList& rhs);

 void insert(const int& itemA);
 /* PROMISES: inserts new node into the list in ascending order*/
 void remove(const int& itemA);
 /* PROMISES: removes the first node in the list that its itemM value is equal to itemA.
    Otherwise it does nothing.*/
 void reverse();
 /* PROMISES: rearranges the nodes of the list in reverse order (descending order)  */

 Node* getHead()const {return headM;}
private:
  Node *headM;
  void insert_first(const int& itemA);
 /* PROMISE: inserts a node object with the itemA to the beginning of the list */
};
```

```
void ExamList::insert(const int& itemA){
  Node *new_node = new Node(itemA);

  if (headM == 0 || itemA <= headM->itemM ) {
    new_node->nextM = headM;
    headM = new_node;
  }
  else {
    Node *before = headM;
    Node *after = headM->nextM;

    while(after != 0 && itemA > after->itemM) {
       before = after;
       after = after->nextM;
    }
     new_node->nextM = after;
     before->nextM = new_node;
  } // end of else
}
```

Part a (3 marks) Write the definition of helper member function **insert_first**, in the following space

```
Node *newNode=new Node;newNode->itemM=itemA;newNode->next=headM;headM=newNode;
```

Part b (4 marks) Write the definition of **copy constructo**r for class `ExamList`, in the following space. You are allowed to use other member functions of the class.

```
ExamList(const ExamList &source){     while(source.headM){          insert(source.itemM);          source.
```

Part c (7 marks) - In the following space, write the definition of function `reverse` that rebuilds the list in a reverse order. You are allowed to use other member functions.

```
Node *temp=headM;while(temp){     temp=temp->next;}while(headM){     headM=temp;     headM=I
```

Part d (6 marks) In the following space write the definition of global function (not a member of class ExamList) called `save_binary` that saves the data values of itemM in each node into a binary file. For further details please read the function interface comment.

```
void save_binary(const ExamList& thelist, const char* filename);
// REQUIRES: the list refers to an existing ExamList object and filename points to a
// valid c-string.
// PROMISES: opens a binary file using filename, and writes the values of itemM from
// each node of the list into the binary file.
```

```
ofstream out(filename, ios::binary);out.write(reinterpret_cast<int*>(thelist.headM->itemM), sizeof(int));ou
```

SECTION VI (10 marks).

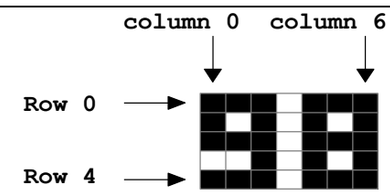
Draw a memory diagram for the point one.

```
int bar[ ] = {16, 25, 36, 49};
const char *func(const char **x, int n){
    int i, j, c, max = 0;
    const char *r = x[0];
    for (i = 0; i < n; i++) {
      c = 0;
      for (j = 0; x[i][j] != '\0'; j++)
          if (x[i][j] == 'a')
              c++;
      if (c > max) {
          max = c;
          r = x[i];
      }
    }
    // point one
    return r;
}
```

```
int main(void)
{
  int *x[2] = {new int[2]};
  **x = x[0][1] = 200;
  x[1] = bar;
  (*x)[1] = *x[1];

  const char *y[] = {
        "banana",
        "quux"
  };
  const char *p = func(y, 2);
  return 0;
}
```

SECTION VII (total of 12 marks)

Computer systems have many different ways to represent images. A very simple approach is to represent the image as a 2-dimensional array of pixels. Below is a tiny image with some white pixels and some black pixels . . .



For each pixel, three numbers between 0 and 255 are used to indicate its colour and brightness. In C++ these types are useful:

```
typedef unsigned char uchar;
struct Pixel { uchar r; uchar g; uchar b; };
```

r, g and b stand for red, green, and blue. On most modern computers, the possible values for unsigned char range from 0 to 255, so it makes sense to use that type for r, g and b. ***(continuous on the next page)***

Here are some example uses of the Pixel type:

```
const Pixel BLACK = {0, 0, 0};
const Pixel BRIGHT_RED = {255, 0, 0};
const Pixel BRIGHT_GREEN = {0, 255, 0};
```

And here is some C++ code that uses the Pixel type to create an Image type:

```
const Pixel MID_GRAY = {128, 128, 128};
const size_t DEFAULT_HEIGHT = 100;
const size_t DEFAULT_WIDTH = 200;

class Image {
public:
    Image(size_t nr = DEFAULT_HEIGHT, size_t nc = DEFAULT_WIDTH);
    Image(const Image& src);
    ~Image();
    Image& operator=(const Image& rhs);
    size_t nrow() const { return nrowM; }
    size_t ncol() const { return ncolM; }
    const Pixel& get_pixel(size_t r, size_t c) const {
        return storeM[ncolM * r + c];
    }
    void set_pixel(size_t r, size_t c, const Pixel& p) {
        storeM[ncolM * r + c] = p;
    }
private:
    size_t nrowM;
    size_t ncolM;
    Pixel *storeM;
};

Image::Image(size_t nr, size_t nc)
: nrowM(nr), ncolM(nc), storeM(new Pixel[nr * nc]){
    for (size_t i = 0; i < nrowM * ncolM; i++)
        storeM[i] = MID_GRAY;
}
```

Part a (6 marks). For the purposes of this problem, let's define the brightness of a pixel to be the sum of its red, green, and blue components, divided by 765.0 to produce a value between 0.0 and 1.0. (Note that 255 + 255 + 255 = 765). Write a C++ function definition to match the given prototype.

```
double max_brightness(const Image& im);
// REQUIRES: im.nrow() > 0 && im.ncol() > 0.
// PROMISES:Return value is the maximum brightness among all the pixels of im.
```

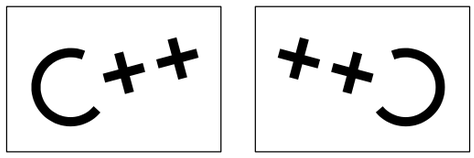
int max=0;for(int i=0; i<(int) im.nrow(); i++){ for(int j=0; j<(int) im.ncol(); j++){ Pixel p=im.get_pixel

Part b (6 marks). For convenience, here is a repeat of the class definition from the previous page:

```
class Image {
public:
    Image(size_t nr = DEFAULT_HEIGHT, size_t nc = DEFAULT_WIDTH);
    Image(const Image& src);
    ~Image();
    Image& operator=(const Image& rhs);
    size_t nrow() const { return nrowM; }
    size_t ncol() const { return ncolM; }
    const Pixel& get_pixel(size_t r, size_t c) const {
        return storeM[ncolM * r + c];
    }
    void set_pixel(size_t r, size_t c, const Pixel& p) {
        storeM[ncolM * r + c] = p;
    }
private:
    size_t nrowM;
    size_t ncolM;
    Pixel *storeM;
};
```

Consider the problem of generating an image B that is the “mirror image" of an original image A, as shown below.
Original:

original image mirror image



Write a C++ function definition to match the given prototype.

```
Image mirror(const Image& im);
// REQUIRES: im.nrow() > 0 && im.ncol() > 0.
// PROMISES: Return value is an Image object that is the mirror image of im.
```

```
Image temp;for(int i=(int) im.nrow(); i>0; i--){     for(int j=(int) im.ncol(); j>0; j--){          temp.set_pixel(i, j,
```

SECTION VIII (total of 8 marks).

Part a (4 marks). A recursive function is not the most efficient way to find the length of a C-style string, but it works unless the string is incredibly long. Draw a memory diagram for the first time the given program reaches point one.

```
int len(const char *s){
    int r = 0;
    if (*s != '\0')
        r = 1 + len(s + 1);
    // point one
    return r;
}
```

```
int main(void)
{
    int k;
    k = len("YZ");
    return 0;
}
```

Part b - In this section you should write the recursive solution of a function called `isPalindromeWord` that returns true if a word is spelled the same from both ends **(4 marks).**

```
bool isPalindromeWord(const char* s, int length);
/* REQUIRES: length > 0, and s points to a word (a sequence of
 * alphabetic characters)
 * PROMISES: returns true if string s is a palindrome word (a word that
 * spells the same from both ends). Otherwise returns false*/
```