CEng 242 Homework 5

Due: 18^{th} May 2001

In this homework you will implement C++ classes for a virtual window management system. You are given the following declarations in the given `screen.h` file:

```
enum Event {LEFTCLICK,RIGHTCLICK,RESIZE,TOTOP,TOBOTTOM,MOVE,CLOSE};

class Screen;

class Window {
        int width,height;
        char name[21];
public:
        Window(int, int, int, int, const char *,Screen *);
        char *getName();
        int getWidth();
        int getHeight();
        virtual void handle(Event ev,int x, int y, int x2=0, int y2=0) {
                switch (ev) {
                case RESIZE:    width=x; height=y;
                                resize(x,y);
                                break;
                case LEFTCLICK: leftclick(x,y);
                                break;
                case RIGHTCLICK:rightclick(x,y);
                }
        }
        virtual void leftclick(int x,int y) { };
        virtual void rightclick(int x,int y) { };
        virtual void resize(int x,int y) { width=x; height=y; };
};

class Screen {
        int height, width;
protected:
        struct WinList {
                Window *win;
                int x,y;
        } winlist[20];
        int n;
public:
        Screen(int, int);
        void addWindow(Window *, int, int);
        virtual void handle(Event ev,int x, int y, int x2=0, int y2=0);
};

class Menu:public Window {
        char item[10][21];
        int n;
```

```
public:
        Menu(int, int, int, int,const char *,Screen *);
        void addItem(const char *s);
        void deleteItem(int j);
        void leftclick(int x, int y);
        void rightclick(int x, int y);
};

class Canvas:public Window {
        char *bitarr;
public:
        Canvas(int, int, int, int,const char *,Screen *);
        ~Canvas();
        void switchPixel(int, int);
        int getPixel(int, int);
        void leftclick(int, int);
        void rightclick(int, int);
void resize(int,int);
};

class NestedScreen:public Window, public Screen {
public:
        NestedScreen(int, int, int, int,const char *,Screen *);
        void handle(Event ev,int x, int y, int x2=0, int y2=0);
};
```

A `Screen` object can update upto 20 window like objects and it keeps track of their placement offsets (`x` and `y` dimensions) and a pointer to them (`win`). A new window is added at the end of the list it is stacked at the topmost position of the screen. Member `n` is the current number of windows on the screen.
Following member functions are defined for the class `Screen`:

`Screen(int w,int h);`
: Constructor setting `width` and `height` of the screen from `w` and `h` respectively.

`void addWindow(Window *w, int x, int y);`
: Inserts the window pointer `w` at the last empty position in the `winlist` array.

`void handle(Event ev, int x, int y, int x2=0, int x2=0)`
: This function scans all windows from top to the last for a window containing the `(x,y)` point and if it finds a match handles the `ev` event only for the first matched window. An event can be any of:

  `TOTOP`
  : Matched window is moved to top (last in the `winlist`) position. `winlist` is modified accordingly.

  `TOBOTTOM`
  : Matched window is moved to bottom (position 0 in the `winlist`). `winlist` is modified accordingly.

  `MOVE`
  : Offset (position in the screen placement) of the matching window is changed to `(x2, y2)` point.

`CLOSE`
: Matched window is deleted from the window list (`winlist`).

`RESIZE`
: A `(RESIZE,x2,y2)` message is sent to matching window by calling the `handle(...)` method of the window.

Other messages
: Simply passed to matched window as `(ev,x',y')` by calling `handle(...)` where `(x',y')` is the point `(x,y)` in window relative coordinates (simply subtraction of window offsets from the `(x,y)` parameters).

Class `Window` stores the window width, height and name. It has the following member functions:

`Window(int x, int y, int w, int h,const char *n,Screen *s);`
: Constructs object with weight `w`, height `h` name `n` and calls `addWindow(...)` method of the screen `s` to insert the constructed window at `(x,y)` offset.

`char *getName();`
: Returns the name of the window.

`int getHeight();`
: Returns the height of the window.

`int getWidth();`
: Returns the width of the window.

Class `Menu` is derived from `Window` and contains a list of menu item names and `n` is the number of items which should be initialized to 0. Constructor makes initializations and passes parameters to the `Window` constructor. It has the following member functions:

`void addItem(const char *s);`
: Adds the menu item with name `s` at the end of the list.

`void deleteItem(int j);`
: Deletes the j^{th} item from the menu list.

`void leftclick(int x, int y);`
: Outputs the message line:
`*WindowName*: (Left) *MenuItem*`
at standart output. `*WindowName*` is the name of the menu window. `*MenuItem*` is the name of the menu item at event point `(x,y)`. If there is no menu item in the list yet it is "`NOMENUITEM`". The active menu item is calculated as the element '`y*n/`height' in the `item` array.

`void rightclick(int x, int y);`
: Same of `leftclick` except the string `(Left)` is replaced by the string `(Right)`.

Class `Canvas` is derived from `Window` and contains a bitmap array where a pixel in the window is represented by a bit in this array. The pixel `(x,y)` is the bit in the position `y*width+x` in the `bitmap` array. Constructor allocates the required space (number of pixels/8 characters) in `bitmap` array, initializes all pixels (bits) to 0 and sends other parameters to `Window` constructor. It has the following member functions:

`~Canvas()`
: Deallocates the space for the bitmap.

`void switchPixel(int x, int y);`
: Reverts the value of the pixel at `(x,y)` position. (0 to 1, 1 to 0)

`int getPixel(int x, int y);`
Returns the pixel value at `(int x, int y)` position. (0 or 1)

`void leftclick(int x, int y);`
Calls `switchPixel(int x, int y)`.

`void rightclick(int x, int y);`
According to value of the pixel `(x,y)` Outputs one of the following message lines at standart output:
WindowName`: Black`
WindowName`: White`
for 1 and 0 respectively.

`void resize(int w, int h);`
Reallocates the storage of the `bitmap` for new size information, `(w,h)`. All pixels in the newly allocated bitmap is reset (set to all 0's).

Class `NestedScreen` is a `Window` which behaves like a small `Screen`. Constructor only passes parameters to its super classes. Following member function exists:

`void handle(Event ev, int x, int y, int x2, int y2);`
If event is one of `TOTOP, TOBOTTOM, MOVE` or `CLOSE` calls the `Screen handle(...)` function with the same parameters. Similarly if `(x,y)` point matches (it is inside of) any window in the window list of the `NestedScreen` it calls the `Screen` handler. If the point is not inside of any window, it calls the `handle(...)` function of the `Window` class with same parameters.

For simplicity there is no error reporting for this homework. Only check for out of bounds and silently ignore the request. Cases are: a request out of screen borders, a request out of window borders, a request for inserting to a full list (more than 20 windows or more than 10 menu items), parts of strings exceeding 20 characters. A window offset should be inside of the screen borders but its bottom-right corner can be outside.
You only need and are asked to implement the unimplemented (not inline) member functions of the given classes in a file. You are not allowed to write any other functions but include any C, C++ header files in file "`screen.cpp`". You should only submit this file.
You can use the following "`Makefile`" in development (spaces in the start of the line following ':' come from a tab):

```
CCC=g++
CPPFLAGS=-g

.cpp.o:
        $(CCC) $(CPPFLAGS) -c $<

hw5: screen.h screen.o hw5.o
        $(CCC) $(CPPFLAGS) -o hw5 screen.o hw5.o
```

"`hw5.cpp`" file should contain the `int main()` function that you will use during development, "`screen.h`" is the header file given above and "`screen.cpp`" contains the member functions that you will implement and submit.
We will post test data as soon as it is available.