OOP344 test One November 1st. V2  
please Print…

Name:

Surname:

Seneca Email id:

Section:

Total 64 marks,

55 gets 100%

#include <iostream>

#include <cstdarg>

// A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

using namespace std;

#define V1

#define mul(a, b) (a\*b)

#define CrtSet(v, t, c) t v = c

int a[3][5] = {{10, 20, 30, 40, 50},

{60, 70, 80, 90, 100},

{110, 120, 130, 140, 150}};

void w1(int n, char\* args[]){

cout<<"W1:"<<endl

<<n<<endl

<<args[2]<<endl

<<a[args[2][0]-'0'][args[2][1]-'0']<<endl;

}

void w2(){

int\* p = a[0]+1;

cout<<"W2:"<<endl

<<p[0]<<endl

<<\*(p+2)<<endl

<<p[10]<<endl;

}

void w3(){

CrtSet(x, int, 4);

CrtSet(y, int, 5);

CrtSet(z, int ,6);

cout<<"W3:"<<endl;

#ifdef V1

x = mul(x+2, y);

#endif

#ifdef V2

x = mul(x, y+2);

#endif

cout<<x<<endl;

}

void w4(void\* p){

int (\*ar)[3] = (int(\*)[3])p;

cout<<"W4:"<<endl<<ar[2][2]<<endl;

}

int w5(int a, int b = 10, int c = 20, int d = 30){

cout<<"W5:"<<endl;

return a+b+c+d;

}

int& w6f(){

static int x = 10;

x--;

return x;

}

void w6(){

cout<<"w6:"<<endl;

for(int i=0;i<5;i++){

cout<<w6f()--<<" ";

}

cout<<endl;

}

void w7(const char\* s, ...){

va\_list vl;

va\_start(vl, s);

int a;

int i=0;

cout<<"w7:"<<endl;

while(a = va\_arg(vl, int)){

cout<<char(s[i++]+a);

}

cout<<"!"<<endl;

va\_end(vl);

}

int main(int argc, char\* argv[]){

w1(argc, argv);

w2();

w3();

w4(a);

cout<<w5(30, 40)<<endl;

w6();

w7("Walking",-19,14,2,-6,0);

return 0;

}

The following questions are done in “oop344” namespace. Create header files and Source files for each independent class.

Question ONE:  
Header: 11 marks  
Source: 7 marks

Having following two structures:

struct Color{

unsigned char \_red;

unsigned char \_green;

unsigned char \_blue;

Color(unsigned char red = 255, unsigned char green = 255,unsigned char blue = 255);

};

struct Coordinates{

int \_x;

int \_y;

Coordinates(int x=0, int y=0);

};

Create an abstract base class called Shape.

Shape has the following attributes:

1 - Shape has Color.   
2 - Shape has Coordinates on the screen.  
3 - Shape has width (number of pixels)  
4 - Shape has height (number of pixels)  
5 - Shape could be solid or not.

Constructors:

* Shape can be created by passing at least Color and Coordinates.   
  If solid is not provided, the shape will be solid.  
  If either of width or height is not provided, it will be defaulted to zero.
* Shape can also be created with no arguments (defaulted) , in which case, Color and Coordinates will be defaulted.

Methods:

* Shape should be able to draw itself. But this method has to be implemented in future descendants and not in Shape itself.
* Shape can hide (or disappear). This method also is yet to be implemented in future descendants and not in shape itself.
* Shape can move.   
  Move will receive new Coordinates and move to it. This is accomplished by first hiding, then setting its own coordinates to the new ones and then redrawing itself.   
  *(No validation is done in this case and if the Shape is moved outside of the screen the results are undefined)*
* Shape can do a validated-move too. A validated move will move the shape to a new location only if it is possible. (i.e. it makes sure the shape can be drawn at new coordinates and will not go off screen)

In this case move not only receives the coordinates, but also it will receive an address to a function that validates the move and makes sure the shape will fit on the screen at the new coordinates. *(please note: since validated-move receives a function pointer to a validation function. You do not need to worry about “HOW” the validation is done. You only need to know how to call the validation function and get a true or false value)*

A move validation function has the following signature.  
bool CanMove(const Shape& S, Coordinates C);  
This function is to receive a Shape and new coordinates and check and make sure the shape can be drawn on the screen.

Validated move will accomplish this by passing itself and the new coordinates to the validation function. If the validation function returns a true value, it will use the “move” method to move, otherwise it will do nothing.

Question Two:

Header: 11 marks  
Sourde: 20 marks

Create a standard stack of strings called StrStack:

*(string: null terminated array of characters)*

The StrStack must be fully compliable with header file and source code file.

Apart from all the necessary constructor and destructor StrStack must have the following methods:

void push(const char\* data);

Pushes a string on unknown length into the stack

(note: dynamic memory allocation in node)

void pop(char\* data);

pops the top string of the stack out and copies the data of the node into the argument “data”

bool isEmpty()const;

returns true of the StrStack is empty;

const char\* operator[](int index)const;   
returns the address of the data of the node corresponding to the index. Where index “0” is the top node in the stack.

If there is no node at the requested stack, the index operator returns NULL.

Your program should be working with following tester program (see next page):

#include <iostream>

#include "StrStack.h"

using namespace std;

using namespace oop344;

// note that this is only a test and the length of the strings

// pushed into the stack are limited only by the size of the memory available.

int main(){

char str[81];

StrStack S;

int i;

S.push("Hello");

S.push("this");

S.push("is");

S.push("a");

S.push("test!");

for(i=0;S[i];i++); // finding the last index

for(i--;i>=0;i--){ // printing in reverse order

cout<<S[i]<<" ";

}

cout<<endl;

while(!S.isEmpty()){

S.pop(str);

cout<<str<<endl;

}

return 0;

}

Output:

**Hello this is a test!  
test!  
a  
is  
this  
Hello**