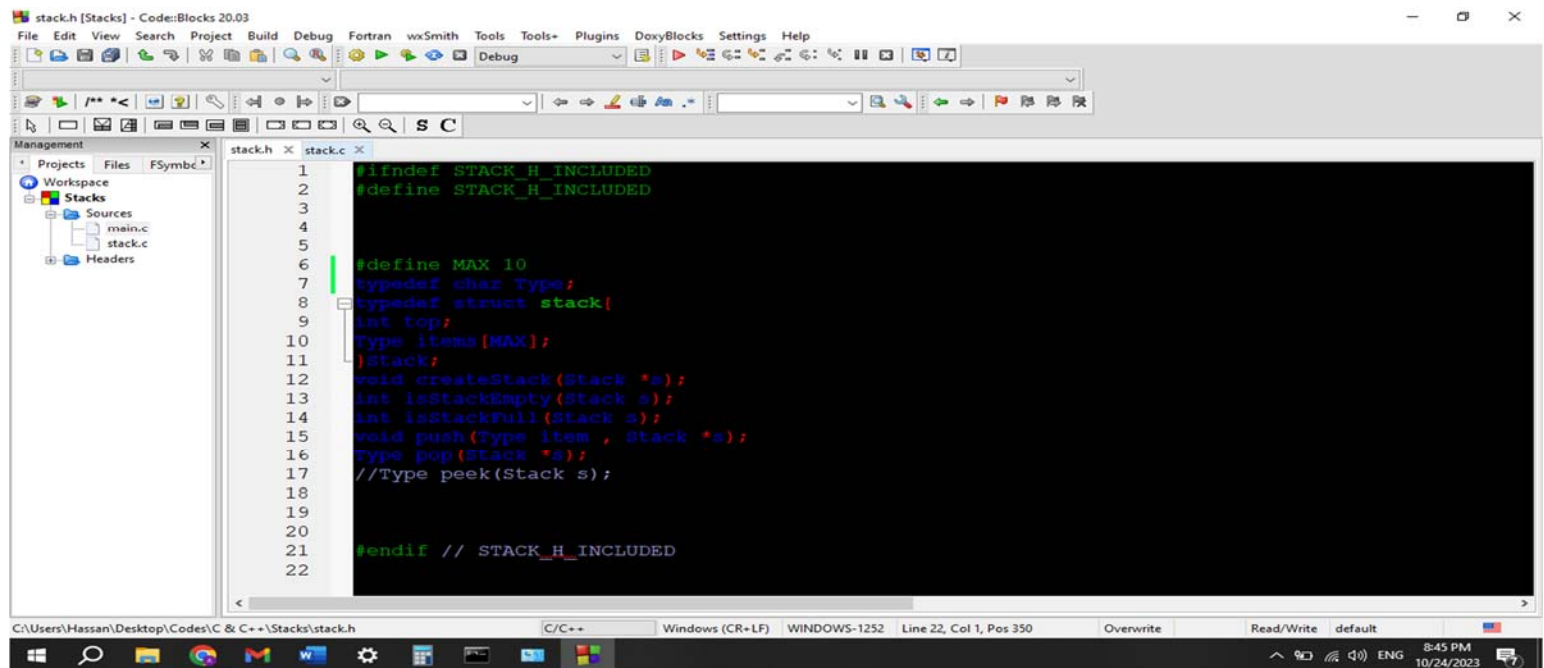


Sheet #1 – Ahmed Hassan Ahmed Emara - 20220017

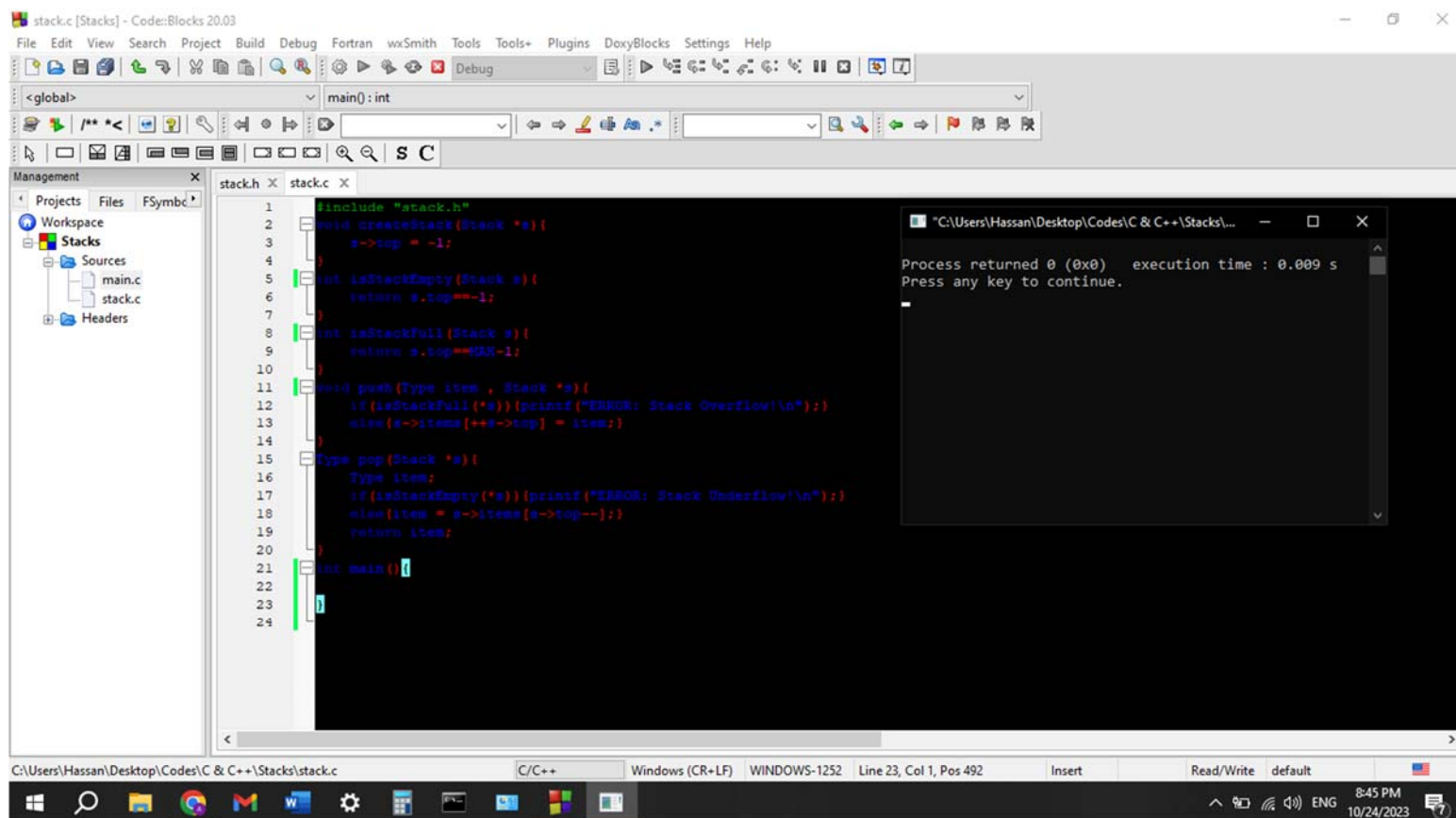
1)



The screenshot shows the Code::Blocks 20.03 IDE with the 'stack.h' file open. The file contains the following code:

```
1 #ifndef STACK_H_INCLUDED
2 #define STACK_H_INCLUDED
3
4
5
6 #define MAX 10
7 typedef char Type;
8 typedef struct stack{
9     Type items[MAX];
10     int top;
11 }Stack;
12 void createStack(Stack *s);
13 int isStackEmpty(Stack s);
14 int isStackFull(Stack s);
15 void push(Type item , Stack *s);
16 Type pop(Stack *s);
17 //Type peek(Stack s);
18
19
20
21 #endif // STACK_H_INCLUDED
22
```

The status bar at the bottom indicates the file path is C:\Users\Hassan\Desktop\Codes\C & C++\Stacks\stack.h, the language is C/C++, and the current position is Line 22, Col 1, Pos 350.



The screenshot shows the Code::Blocks 20.03 IDE with the 'stack.c' file open. The file contains the following code:

```
1 #include "stack.h"
2 void createStack(Stack *s){
3     s->top = -1;
4 }
5 int isStackEmpty(Stack s){
6     return s.top == -1;
7 }
8 int isStackFull(Stack s){
9     return s.top == MAX-1;
10 }
11 void push(Type item , Stack *s){
12     if(isStackFull(s)){printf("ERROR: Stack Overflow\n");}
13     else{s->items[++s->top] = item;}
14 }
15 Type pop(Stack *s){
16     Type item;
17     if(isStackEmpty(s)){printf("ERROR: Stack Underflow\n");}
18     else{item = s->items[s->top--];}
19     return item;
20 }
21 int main(){
22
23
24 }
```

The status bar at the bottom indicates the file path is C:\Users\Hassan\Desktop\Codes\C & C++\Stacks\stack.c, the language is C/C++, and the current position is Line 23, Col 1, Pos 492.

On the right side, a terminal window shows the execution output:

```
"C:\Users\Hassan\Desktop\Codes\C & C++\Stacks\..."
Process returned 0 (0x0)   execution time : 0.009 s
Press any key to continue.
```

Yes, in order to make stack.c able to build functions declared in stack.h .

2)

The screenshot displays the Code::Blocks IDE interface. The main editor window shows a C program named `test.c` with the following code:

```
1 #include<stdio.h>
2 #include"stack1.h"
3 int main()
4 {
5     Stack s;
6     char choice;
7     type element;
8     printf("(a) Read an element then Push it.\n");
9     printf("(b) Pop an element then displays it.\n");
10    printf("(c) Exit.\n");
11    scanf("%c",&choice);
12    if(choice=='a'){
13        printf("Enter the element: ");
14        scanf("%d",&element);
15        push(element, &s);
16    }
17    else if(choice=='b'){
18        printf("%d",pop(&s));
19    }
20    else{return 0;}
21 }
22
```

The left sidebar shows the project structure for `sheet1`, including `Sources` (containing `stack1.c` and `test.c`) and `Headers` (containing `stack1.h`).

An output window titled `"C:\Users\Hassan\Desktop\Codes\C & C++\sheet1..."` displays the program's execution results:

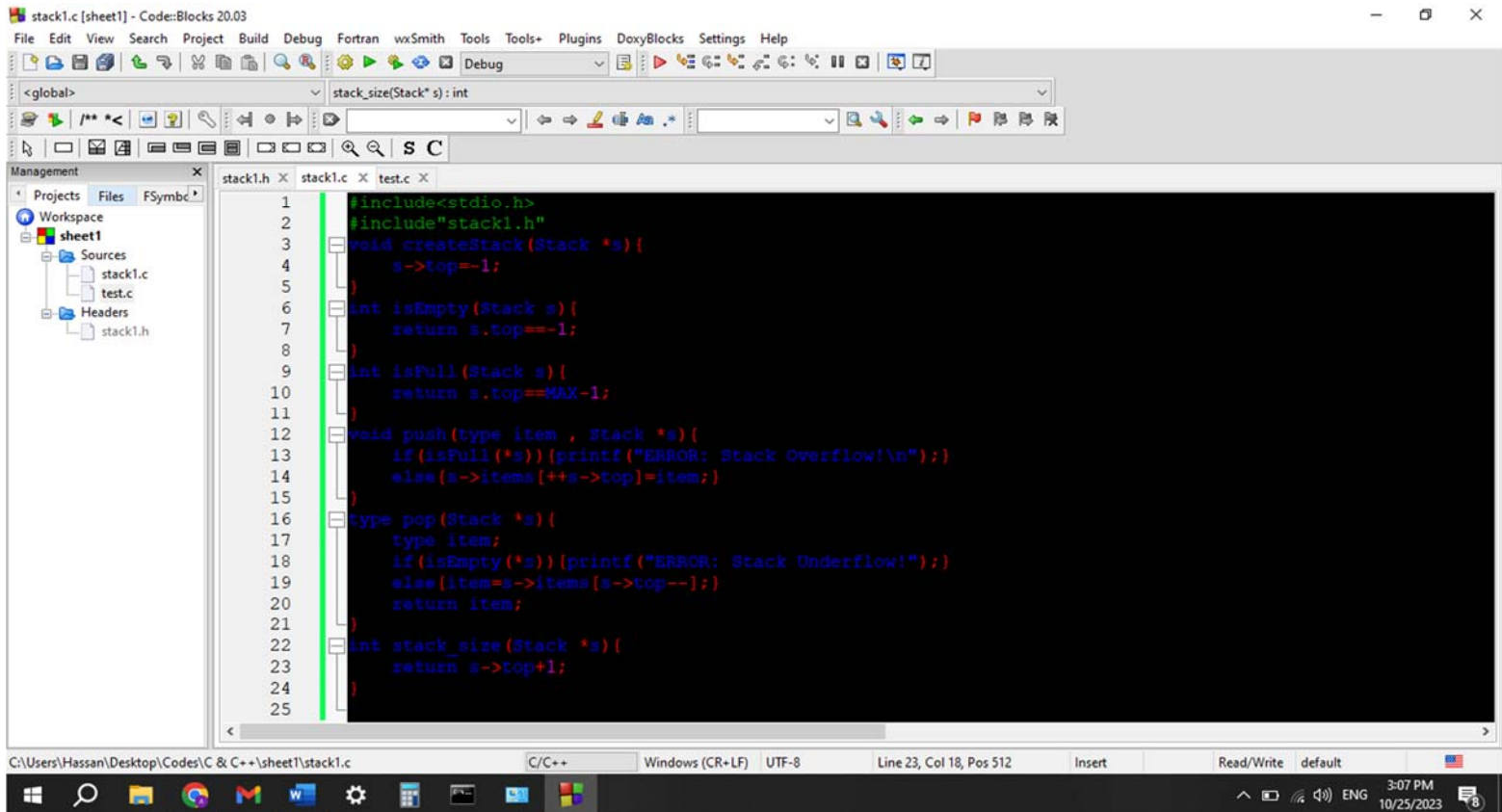
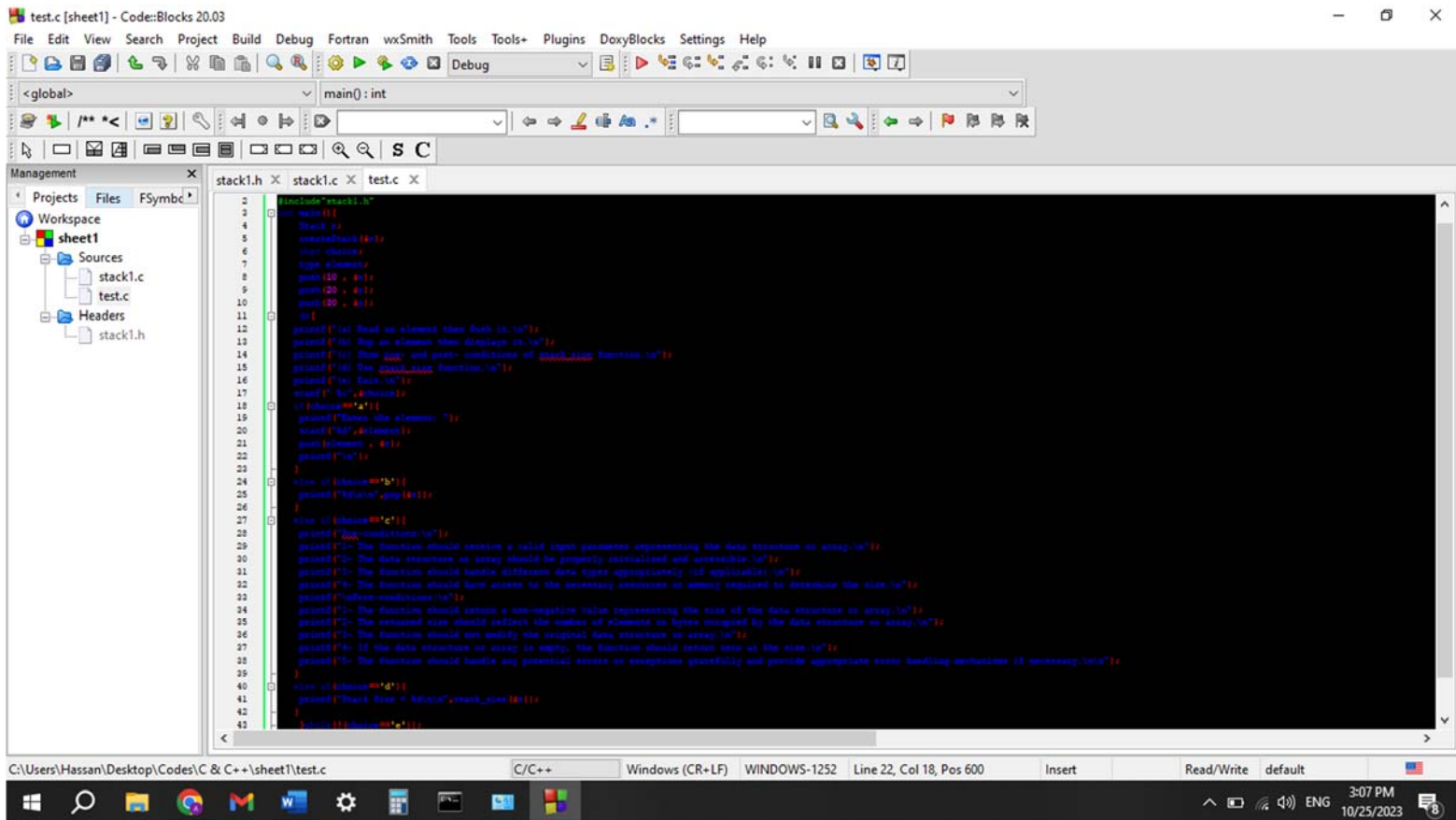
```
(a) Read an element then Push it.
(b) Pop an element then displays it.
(c) Exit.
a
Enter the element: 1

Process returned 0 (0x0)   execution time : 4.359 s
Press any key to continue.
```

The status bar at the bottom indicates the file path `C:\Users\Hassan\Desktop\Codes\C & C++\sheet1\test.c`, the language `C/C++`, and the current position `Line 22, Col 1, Pos 459`.

Yes, because it is executable.

3)



4)

The screenshot shows a C++ IDE with a project named 'sheet1'. The code is organized into three files: 'stack1.h', 'stack1.c', and 'test.c'. The 'test.c' file is currently open and displays the following code:

```
1 #include<stdio.h>
2 #include"stack1.h"
3 void content(Stack s){
4     int item;
5     while(!isEmpty(s)){
6         printf("%d\n",pop(&s));
7     }
8 }
9 int main(){
10     Stack s;
11     createStack(&s);
12     type item;
13     push(10 , &s);
14     push(20 , &s);
15     push(30 , &s);
16     content(s);
17 }
18
```

The output window on the right shows the execution results:

```
30
20
10

Process returned 0 (0x0)   execution time : 0.097 s
Press any key to continue.
```

The status bar at the bottom indicates the current file is 'test.c', the language is 'C/C++', and the cursor is at 'Line 18, Col 1, Pos 291'. The system clock shows 5:27 PM on 10/25/2023.

Because we are supposed to know nothing about stack.c .

5), 6)

```
type first(Stack s){
    if(isEmpty(s)){printf("ERROR: Stack Is Empty!");}
    else{return s.items[0];}
}

type pop_value(Stack s){
    type item;
    if(isEmpty(s)){printf("ERROR: Stack Underflow!");}
    else{item=s.items[s.top--];}
    return item;
}

type last(Stack s){
    return pop_value(s);
}
```

The screenshot shows the Code::Blocks IDE with the following components:

- Project Explorer:** Shows a project named 'sheet1' with source files 'stack1.c', 'test.c', and 'stack1.h'.
- Editor:** Displays the code for 'test.c' with line numbers 2 to 19. The code includes 'stack1.h' and implements a 'main' function that creates a stack, pushes values 10, 20, and 30, and then prints the first element and the popped value.
- Output Console:** Shows the execution results: '10', '30', 'Process returned 0 (0x0) execution time : 0.078 s', and 'Press any key to continue.'.
- Status Bar:** Indicates the current file is 'C:\Users\Hassan\Desktop\Codes\C & C++\sheet1\test.c' and the cursor is at Line 18, Col 2, Pos 340.

```
2 #include "stack1.h"
3 /*void content(Stack s){
4     int item;
5     while(!isEmpty(s)){
6         printf("%d\n", pop(&s));
7     }
8 }*/
9 int main(){
10     Stack s;
11     createStack(&s);
12     type item;
13     push(10 , &s);
14     push(20 , &s);
15     push(30 , &s);
16     printf("%d\n", first(s));
17     printf("%d\n", pop_value(s));
18 }
19
```

Output Console:

```
10
30
Process returned 0 (0x0) execution time : 0.078 s
Press any key to continue.
```

7)

```
void destroy(Stack *s){
    type item;
    while (!isEmpty(*s)) {
        item=s->items[s->top--];
    }
}
```

```
int main(){
    Stack s;
    createStack(&s);
    type item;
    push(10 , &s);
    push(20 , &s);
    push(30 , &s);
    destroy(&s);
    printf("\nStack Has Been Destroyed Successfully!\n");
}
```

"C:\Users\Hassan\Desktop\Codes\C & C++\she... — □ ×

Stack Has Been Destroyed Successfully!

Process returned 0 (0x0) execution time : 0.016 s
Press any key to continue.

8)

```
void cpy(Stack s){
    Stack x;
    createStack(&x);
    type arr[MAX];
    int counter=0;
    while(!isEmpty(s)){
        arr[counter]=pop(&s);counter++;
    }
    counter=0;
    while(!isFull(x)){
        push(arr[counter] , &x);counter++;
    }
    counter=0;
    while(!(counter==MAX)){
        printf("%d\n" , x.items[counter]);counter++;
    }
}
```

The screenshot displays the Code::Blocks IDE interface. The main editor window shows the implementation of the `cpy` function and the `main` function. The `main` function pushes values from 10 to 100 onto a stack `s` and then calls `cpy(s)`. The output window shows the copied values from 100 down to 10. The status bar at the bottom indicates the current file is `test.c` and the cursor is at line 16, column 12, position 295.

```
#include<stdio.h>
#include"stack1.h"
int main(){
    Stack s;
    createStack(&s);
    type item;
    push(10 , &s);
    push(20 , &s);
    push(30 , &s);
    push(40 , &s);
    push(50 , &s);
    push(60 , &s);
    push(70 , &s);
    push(80 , &s);
    push(90 , &s);
    push(100 , &s);
    cpy(s);
}
```

Process returned 0 (0x0) execution time : 0.042 s
Press any key to continue.

9)

```
int stack_size(Stack *s){
    return s->top+1;
}
```

The screenshot shows the Code::Blocks IDE with a C program being executed. The program defines a stack structure and a function to push elements. In the main function, 10 elements are pushed onto the stack, and then the stack size is printed using the `stack_size` function. The output window displays the result: "Stack Size = 10".

```
#include "stack1.h"

int main() {
    Stack s;
    createStack(&s);
    type item;
    push(10, &s);
    push(20, &s);
    push(30, &s);
    push(40, &s);
    push(50, &s);
    push(60, &s);
    push(70, &s);
    push(80, &s);
    push(90, &s);
    push(100, &s);
    printf("Stack Size = %d", stack_size(&s));
}
```

Output: Stack Size = 10

10)

The screenshot shows the Code::Blocks IDE with a C program being executed. The program defines a stack structure and a function to push elements. In the main function, 10 elements are pushed onto the stack, and then the first element is printed using the `fst` function. The output window displays the result: "First Element: 10".

```
#include <stdio.h>
#include "stack1.h"

type fst(Stack s){
    type x;
    while(!isEmpty(s)){
        x=pop(&s);
    }
    return x;
}

int main(){
    Stack s;
    createStack(&s);
    type item;
    push(10, &s);
    push(20, &s);
    push(30, &s);
    push(40, &s);
    push(50, &s);
    push(60, &s);
    push(70, &s);
    push(80, &s);
    push(90, &s);
    push(100, &s);
    printf("First Element: %d\n", fst(s));
}
```

Output: First Element: 10

11)

test.c [sheet1] - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> list(Stack s): type

Management

Projects Files FSymbc

Workspace

sheet1

Sources

stack1.c

test.c

Headers

stack1.h

```

1 #include<stdio.h>
2 #include"stack1.h"
3 type list(Stack s){
4     type x;
5     x=pop(&s);
6     push(x , &s);
7     return x;
8 }
9
10 int main() {
11     Stack s;
12     createStack(&s);
13     type item;
14     push(10 , &s);
15     push(20 , &s);
16     push(30 , &s);
17     push(40 , &s);
18     push(50 , &s);
19     push(60 , &s);
20     push(70 , &s);
21     push(80 , &s);
22     push(90 , &s);
23     push(100 , &s);
24     printf("Last Element: %d\n",list(s));
25 }

```

"C:\Users\Hassan\Desktop\Codes\C & C++\sheet1..."

Last Element: 100

Process returned 0 (0x0) execution time : 0.015 s

Press any key to continue.

C:\Users\Hassan\Desktop\Codes\C & C++\sheet1\test.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 7, Col 14, Pos 120 Insert Read/Write default 3:06 PM 10/26/2023

12)

test.c [sheet1] - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> main(): int

Management

Projects Files FSymbc

Workspace

sheet1

Sources

stack1.c

test.c

Headers

stack1.h

```

1 #include<stdio.h>
2 #include"stack1.h"
3 void dest(Stack *s){
4     type item;
5     while(!isEmpty(*s)){
6         item=s->items[s->top--];
7     }
8 }
9
10 int main() {
11     Stack s;
12     createStack(&s);
13     type item;
14     push(10 , &s);
15     push(20 , &s);
16     push(30 , &s);
17     push(40 , &s);
18     push(50 , &s);
19     push(60 , &s);
20     push(70 , &s);
21     push(80 , &s);
22     push(90 , &s);
23     push(100 , &s);
24     dest(&s);
25     printf("\nStack Has Been Destroyed Successfully!\n");
26 }

```

"C:\Users\Hassan\Desktop\Codes\C & C++\sheet1..."

Stack Has Been Destroyed Successfully!

Process returned 0 (0x0) execution time : 0.035 s

Press any key to continue.

C:\Users\Hassan\Desktop\Codes\C & C++\sheet1\test.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 25, Col 2, Pos 486 Insert Read/Write default 3:15 PM 10/26/2023

13)

```

1 #include<stdio.h>
2 #include"stack1.h"
3
4 void copy(Stack s){
5     Stack s1;
6     createStack(s1);
7     type arr[100];
8     int counter=0;
9     while(!isEmpty(s)){
10         arr[counter]=pop(s);counter++;
11     }
12     counter=0;
13     while(!isFull(s1)){
14         push(arr[counter], s1);counter++;
15     }
16     counter=0;
17     while((counter<=100)){
18         printf("%d\n", s1.item[counter]);counter++;
19     }
20 }
21
22 int main(){
23     Stack s;
24     createStack(s);
25     type item;
26     push(10, s);
27     push(20, s);
28     push(30, s);
29     push(40, s);
30     push(50, s);
31     copy(s);

```

Process returned 0 (0x0) execution time : 0.016 s
Press any key to continue.

14)

```

1 #include<stdio.h>
2 #include"stack1.h"
3
4 int SIZE(Stack s){
5     int counter=0;
6     while(!isEmpty(s)){
7         pop(s);counter++;
8     }
9     return counter;
10 }
11
12 int main(){
13     Stack s;
14     createStack(s);
15     type item;
16     push(10, s);
17     push(20, s);
18     push(30, s);
19     push(40, s);
20     push(50, s);
21     push(60, s);
22     push(70, s);
23     push(80, s);
24     push(90, s);
25     push(100, s);
26     printf("Stack Size = %d\n",SIZE(s));

```

Stack Size = 10
Process returned 0 (0x0) execution time : 0.025 s
Press any key to continue.

15)

```
#include<stdio.h>
#include"stack1.h"
void content(Stack s){
    int item;
    while(!isEmpty(s)){
        printf("%d\n",pop(&s));
    }
}
int main(){
    Stack s;
    createStack(&s);
    type item;
    push(10 , &s);
    push(20 , &s);
    push(30 , &s);
    content(s);
}
```

30
20
10
Process returned 0 (0x0) execution time : 0.097 s
Press any key to continue.

Due to abstraction and encapsulation.

16)

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define MAX_SIZE 100
6
7  // Structure for stack
8  typedef struct {
9      char arr[MAX_SIZE];
10     int top;
11 } Stack;
12
13 // Function to initialize stack
14 void initialize(Stack* stack) {
15     stack->top = -1;
16 }
17
18 // Function to check if stack is empty
19 bool isEmpty(Stack* stack) {
20     return stack->top == -1;
21 }
22
23 // Function to push an element onto the stack
24 void push(Stack* stack, char character) {
25     stack->arr[++stack->top] = character;
26 }
27
28 // Function to pop an element from the stack
29 char pop(Stack* stack) {
30     if (!isEmpty(stack)) {
31         return stack->arr[stack->top--];
32     }
33     return '\0';
34 }
35
36 // Function to check balanced parentheses
37 bool isBalanced(const char* parentheses) {
38     Stack stack;
39     initialize(&stack);
40     int i = 0;
41     while (parentheses[i] != '\0') {
42         if (parentheses[i] == '(' || parentheses[i] == '[' || parentheses[i] == '{') {
43             push(&stack, parentheses[i]);
44         } else if (parentheses[i] == ')' || parentheses[i] == ']' || parentheses[i] == '}') {
45             if (isEmpty(&stack)) {
46                 return false; // If stack is empty and we encounter a closing parenthesis, it's unbalanced
47             }
48             char popped = pop(&stack); // Pop the last opening parenthesis from the stack
49             if ((popped == '(' && parentheses[i] != ')') ||
50                 (popped == '[' && parentheses[i] != ']') ||
51                 (popped == '{' && parentheses[i] != '}')) {
52                 return false; // If the opening and closing parentheses don't match, it's unbalanced
53             }
54         }
55         i++;
56     }
57     return !isEmpty(&stack); // If the stack is empty at the end, it's balanced
58 }
59
60 int main() {
61     char parentheses[] = "({[(){}])";
62     printf("Test 1: %s is %s\n", parentheses, isBalanced(parentheses) ? "Balanced" : "Not Balanced");
63
64     parentheses = "([{}])";
65     printf("Test 2: %s is %s\n", parentheses, isBalanced(parentheses) ? "Balanced" : "Not Balanced");
66
67     parentheses = "({[(){}])";
68     printf("Test 3: %s is %s\n", parentheses, isBalanced(parentheses) ? "Balanced" : "Not Balanced");
69
70     parentheses = "([{}])";
71     printf("Test 4: %s is %s\n", parentheses, isBalanced(parentheses) ? "Balanced" : "Not Balanced");
72
73     parentheses = "({[(){}])";
74     printf("Test 5: %s is %s\n", parentheses, isBalanced(parentheses) ? "Balanced" : "Not Balanced");
75
76     return 0;
77 }

```

```

28  #include <stdio.h>
29  #include <stdlib.h>
30  #include <string.h>
31
32  #define MAX_SIZE 100
33
34  // Structure for stack
35  typedef struct {
36      char arr[MAX_SIZE];
37      int top;
38 } Stack;
39
40 // Function to initialize stack
41 void initialize(Stack* stack) {
42     stack->top = -1;
43 }
44
45 // Function to check if stack is empty
46 bool isEmpty(Stack* stack) {
47     return stack->top == -1;
48 }
49
50 // Function to push an element onto the stack
51 void push(Stack* stack, char character) {
52     stack->arr[++stack->top] = character;
53 }
54
55 // Function to pop an element from the stack
56 char pop(Stack* stack) {
57     if (!isEmpty(stack)) {
58         return stack->arr[stack->top--];
59     }
60     return '\0';
61 }
62
63 // Function to check balanced parentheses
64 bool isBalanced(const char* parentheses) {
65     Stack stack;
66     initialize(&stack);
67     int i = 0;
68     while (parentheses[i] != '\0') {
69         if (parentheses[i] == '(' || parentheses[i] == '[' || parentheses[i] == '{') {
70             push(&stack, parentheses[i]);
71         } else if (parentheses[i] == ')' || parentheses[i] == ']' || parentheses[i] == '}') {
72             if (isEmpty(&stack)) {
73                 return false; // If stack is empty and we encounter a closing parenthesis, it's unbalanced
74             }
75             char popped = pop(&stack); // Pop the last opening parenthesis from the stack
76             if ((popped == '(' && parentheses[i] != ')') ||
77                 (popped == '[' && parentheses[i] != ']') ||
78                 (popped == '{' && parentheses[i] != '}')) {
79                 return false; // If the opening and closing parentheses don't match, it's unbalanced
80             }
81         }
82         i++;
83     }
84     return !isEmpty(&stack); // If the stack is empty at the end, it's balanced
85 }
86
87 int main() {
88     char parentheses[] = "({[(){}])";
89     printf("Test 1: %s is %s\n", parentheses, isBalanced(parentheses) ? "Balanced" : "Not Balanced");
90
91     parentheses = "([{}])";
92     printf("Test 2: %s is %s\n", parentheses, isBalanced(parentheses) ? "Balanced" : "Not Balanced");
93
94     parentheses = "({[(){}])";
95     printf("Test 3: %s is %s\n", parentheses, isBalanced(parentheses) ? "Balanced" : "Not Balanced");
96
97     parentheses = "([{}])";
98     printf("Test 4: %s is %s\n", parentheses, isBalanced(parentheses) ? "Balanced" : "Not Balanced");
99
100    parentheses = "({[(){}])";
101    printf("Test 5: %s is %s\n", parentheses, isBalanced(parentheses) ? "Balanced" : "Not Balanced");
102
103    return 0;
104 }

```

Process returned -1073741819 (0xC0000005) execution time : 2.640 s
Press any key to continue.