PROGRAMOWANIE ZORIENTOWANE OBIEKTOWO

ZADANIA

Z1. Uruchom i przeanalizuj poniższy program.

```
#include <stdio.h>
int main(void) {
	int n, i, x[10];
	scanf("%d", &n);
	for (i = 0; i < n; i++) {
		scanf("%d", &x[i]);
	}
	for (i--; i >= 0; i--) {
		printf("%d", x[i]);
	}
	printf("\n");
	return 0;
}
```

Z2. Uruchom i przeanalizuj poniższy program.

```
#include <stdio.h>
typedef int Bool;
#define MAX 100
int Top = MAX, Stack[MAX];
void push(int e) {
	Stack[--Top] = e;
	return;
}
int pop() {
	return Stack[Top++];
}
Bool empty() {
	return Top == MAX;
}
int main(void) {
	int n, i, x;
	scanf("%d", &n);
	for (i = 0; i < n; i++) {
		scanf("%d", &x);
		push(x);
	}
	for (i--; i >= 0; i--) {
		printf("%d", pop());
	}
	printf("\n");
	printf("Empty:%d", empty());
	printf("\n");
	return 0;
}
```

Z3. Uruchom i przeanalizuj poniższy program.

```
#include <stdio.h>
int main(void) {
	struct Elem {
		int Id;
		int PLN;
		int Persn;
	};
	typedef struct Elem Student;
	int j;
	Student S[] = { 10, 4200, 7,
			20, 3000, 3,
			30, 3000, 2,
			40, 2200, 2 };
	for (j = 0; !(S[j].PLN / S[j].Persn > 900); j++)
		;
	printf("%d \n", S[j].Id);
	return 0;
}
```

Z4. Uruchom i przeanalizuj poniższy program.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Elem Item;
struct Elem { int Val;
		Item* Next;
		};
Item* First;
#define ElemS sizeof(Item)
void push(int e) {
	Item* p;
	p = malloc(ElemS);
	p->Val = e;
	(*p).Next = First;
	First = p;
	return;
}
int pop() {
	Item* p; int e;
	p = First;
	First = (*p).Next;
	e = (*p).Val;
	free(p);
	return e;
}
int main(void) {
	int n, i, x;
	scanf("%d", &n);
	for (i = 0; i < n; i++) {
		scanf("%d", &x);
		push(x);
	}
	for (i--; i >= 0; i--) {
		printf("%d", pop());
	}
	printf("\n");
	return 0;

}
```

Z5. (Python) Uruchom i przeanalizuj poniższy program.

```
import sys
MAX= 100
class Stack:
   def __init__(self):
       self.S= [0]*MAX
       self.Top= MAX
   def Push(self, e):
       self.Top-= 1
       self.S[self.Top]= e
   def Pop(self):
       e= self.S[self.Top]
       self.Top+= 1
       return e
   def Empty(self):
       return self.Top == MAX

s= Stack()
for line in sys.stdin:
   for x in line.split():
      s.Push(x)
while not s.Empty():
     print(s.Pop()," ",end="")
```

Z6. (C++) Uruchom i przeanalizuj poniższy program.

```
#include <cstdio>

typedef int Bool;
#define MAX 100
class Stack {
  int Top, S[MAX];
public:
  void push(int e) {
     S[--Top] = e;
     return;
  }
  int pop() {
     return S[Top++];
  }
  Bool empty() {
    return Top == MAX;
  }
  Stack() {
    Top = MAX;
  }
};
int main() {
       int x;
       Stack s = Stack();
       while (scanf("%d", &x) != EOF) {
              s.push(x);
       }
       while (!s.empty()) {
              printf("%d ", s.pop());
       }
       printf("\n");

}
```

Z7. (C++) Uruchom i przeanalizuj poniższy program.

```
#include <cstdio>
typedef int Bool;
#define MAX 100
class Stack {
protected:
  int  Top, S[MAX];
public:
  void push(int e) {
    S[--Top] = e;
    return;
  }
  int  pop() {
    return S[Top++];
  }
  Bool empty() {
    return Top == MAX;
  }
  Stack() {
    Top = MAX;
  }
};
class WindowStack : public Stack {
protected:
  int Window;
public:
  void Start() {
    Window = Top;
    return;
  }
  int  LookUp() {
    return S[Window++];
  }
  Bool Bottom() {
    return Window == MAX;
  }
  WindowStack() {
    return;
  }
};

int main()
{
  int x;
  WindowStack s = WindowStack();
  while (scanf("%d", &x) != EOF)
    s.push(x);
  s.Start();
  printf("%d  ", s.LookUp());
  printf("%d\n", s.LookUp());
  while (!s.empty())
    printf("%d  ", s.pop());
  printf("\n");
}
```

Z8. (Python) Uruchom i przeanalizuj poniższy program.

```
import sys
MAX= 100
class Stack:
    def __init__(self):
        self.S= [0]*MAX
        self.Top= MAX
    def Push(self, e):
        self.Top-= 1
        self.S[self.Top]= e
    def Pop(self):
        e= self.S[self.Top]
        self.Top += 1
        return e
    def Empty(self):
        return self.Top == MAX
class WindowStack(Stack):
    def __init__(self):
        super().__init__()
        self.Window= 0
    def Start(self):
        self.Window= self.Top
    def LookUp(self):
        TMP= self.S[self.Window]
        self.Window += 1
        return TMP
    def Bottom(self):
        return self.Window == MAX
s= WindowStack()
for line in sys.stdin:
    for x in line.split():
        s.Push(x)
s.Start()
print(s.LookUp())
print(s.LookUp())
while not s.Empty():
    print(s.Pop()," ",end="")
```

Z9. Uruchom i przeanalizuj poniższy program.

```
#include <stdio.h>
int main(void){
       int A, B, *Adr;
       A=3;
       B=4;
       Adr= &A; // *Adr == A
       B= B + *Adr;
       printf("%d %d\n", A, B);
}
```

Z10. (Python) Uruchom i przeanalizuj poniższy program.

```
L=[1990, 1984, 1879]
index = len(L)-1
L.insert(index, 2022)
L.pop(index)
L.remove(1984)
L.sort()
L.reverse()
for i in L:
       print(i)
```

Z11. (Python) Uruchom i przeanalizuj poniższy program.

```
import sys
data= []
max= 0
for line in sys.stdin:
    i, g = line.split()
    i=int(i)
    g=int(g)
    data.append({"Id": i, "Grade": g})
    if g > max:
        max=g
for item in data:
    if item["Grade"] == max:
        print(item["Id"], item["Grade"])
```

Z12. Uruchom i przeanalizuj poniższy program.

```
#include <stdio.h>
int main(void){
       struct item {int Id;
              int Grade;};
       struct item data[100];
       int max, k, i, g, j;
       max= 0;
       k= 0;
       while (scanf("%d %d", &i, &g) != EOF){
              data[k].Id= i;
              data[k].Grade= g;
              if (g > max){
              max= g; }
              k+= 1;
       }
       for (j= 0; j < k; j+= 1){
              if (data[j].Grade == max)
              printf("%d %d\n", data[j].Id, data[j].Grade);
       }
}
```