1. Aşağıdaki programların çıktısı nedir? Programlar içerisinde hata varsa hatanın sebebini yazınız.

```
#include<iostream>
using namespace std;
#include "Stack.h"

int main(int argc,char** argv)
{
	Stack<int> Stackint;

	for(int i=0;i<10;i++)
		Stackint.push(i+0.1);
    cout<<Stackint.sizeofStackInBytes();
	int temp;

	while(Stackint.pop(temp))
		cout<<temp;

}
```

```
#include<iostream>
using namespace std;
#include "Stack.h"

int main(int argc,char** argv)
{
	Stack<int> Stackint;

	int a= 123456789;
	while(a){
		Stackint.push(a%10);
		a=a/10;
	}
	int temp;
	while(Stackint.pop(temp))
		cout<<temp;
}
```

```
#include<iostream>
using namespace std;
#include "Stack.h"

int main(int argc,char** argv)
{
	Stack<char> StackChar;

	for(int i=0;i<10;i++)
		StackChar.push('A'+i);
 cout<< StackChar.sizeofStackInBytes();


	char temp;
	while(StackChar.pop(temp))
	{
		if(temp=='A'+4);
			StackChar.push('a');
		cout<<temp;
	}

}
```

```
#include<iostream>
using namespace std;
#include "Stack.h"

int main(int argc,char** argv)
{
	Stack<char*> StackStr;

	StackStr.push("Merhaba");
	StackStr.push("Nasılsın");
	StackStr.push("Deneme");
	cout<<StackStr.sizeofStackInBytes();

	char *temp;

	while(StackStr.pop(temp))
		cout<<temp<<endl;

}
```

2. Aşağıdaki C++ kodları derlendiğinde ekran çıktısı ne olacaktır. Eğer kodlar içerisinde hata varsa sebebini yazınız

```
#include<iostream>
using namespace std;
#include "Stack.h"
int main(int argc,char** argv)
{
	Stack<int*> StackIntPtr;

	int *pDizi = new int[10];
	StackIntPtr.push(pDizi);

	delete[] pDizi;
	int *temp;
	while(StackIntPtr.pop(temp));

	cout<<temp;
	temp[0] = 10;
	cout<<temp[0];

}
```

```
void ekle(Stack<int> stackInt)
{
	stackInt.push(120);
}

int main(int argc,char** argv)
{
	Stack<int> StackInt;

	ekle(StackInt);

	if(StackInt.isEmpty())
		cout<<"bos";
	else
		cout<<"dolu";

}
```

```
#include<iostream>
using namespace std;
#include "Stack.h"

void ekle(Stack<int>* stackInt)
{
	stackInt->push(120);
}

int main(int argc,char** argv)
{
	Stack<int> StackInt;

	ekle(&StackInt);

	if(StackInt.isEmpty())
		cout<<"bos";
	else
		cout<<"dolu";

}
```

```
#include<iostream>
using namespace std;
#include "Stack.h"

void ekle(Stack<int>& stackInt)
{
	stackInt.push(120);
}

int main(int argc,char** argv)
{
	Stack<int> StackInt;

	ekle(StackInt);

	if(StackInt.isEmpty())
		cout<<"bos";
	else
		cout<<"dolu";

}
```

```
#include<iostream>
using namespace std;
#include "Stack.h"

template<typename T>
void ekle(Stack<T>& s)
{
	T temp;
	s.pop(temp);
	cout<<temp;
}
int main(int argc,char** argv)
{
	Stack<int> StackInt;
	Stack<char*> StackCharPtr;
	Stack<float> StackFloat;

	StackInt.push(11);
	StackCharPtr.push("Merhaba");
	StackFloat.push(2.3f);

	ekle(StackInt);
	ekle(StackFloat);
	ekle(StackCharPtr);
}
```

```
#include<iostream>
using namespace std;
#include "Stack.h"

template<typename T>
void ekle(Stack<T>& s)
{
	cout<<T.sizeofStackInBytes();
}

int main(int argc,char** argv)
{
	Stack<int> StackInt;
	Stack<char*> StackCharPtr;
	Stack<float> StackFloat;

	StackInt.push(11);
	StackCharPtr.push("Merhaba");
	StackFloat.push(2.3f);

	ekle(StackInt);
	ekle(StackFloat);
	ekle(StackCharPtr);
}
```

3. *Stack* şablon sınıfının eleman sayısı sabit olacak şekilde tasarlanmıştır. *Stack* sınıfını genişleyebilecek şekilde yeniden tasarlayınız.
   a. Sınıf içerisine *void grow(int size)* fonksiyonu eklenmelidir. Bu fonksiyon her çağrıldığında Yığına ***size*** kadar hücre eklenecektir. (Önceki veriler kaybedilmemelidir.)
   b. *Stack* sınıfı maksimum hücre sayısını tutacak bir değişkene ihtiyaç duyacaktır.
   c. Yığının hücreleri statik dizi olarak değil dinamik dizi olarak oluşturulmalıdır. (dizi *new* kullanarak oluşturulmalı)
   d. *push* fonksiyonu yığını dolu gördüğünde *false* dönmek yerine *grow* fonksiyonunu çağırıp yığını genişletmelidir
   e. *Stack* sınıfının yok edici fonksiyonunda *heap* hafızasından alınan alanlar (*new* ile) serbest bırakılmalıdır (*delete []* ile)

4. Öz yinelemeli fonksiyonlar iç içe fonksiyon çağrıları yaptıklarından döngü ile bulunan çözümlere göre daha yavaş çalışmaktadırlar. Fonksiyon çağrısı pahalı bir işlemdir. İşlemci vakti almaktadır.
    a. Öz yinelemeli her fonksiyon yığın veri yapısı kullanılarak çözülebilir. Örneğin aşağıdaki faktöriyel fonksiyonun yığın versiyonu verilmiştir.

```
int f(int sayi)
{
       if(sayi==1) return 1;
       return sayi*f(sayi-1);
}
```

```
int f_stack(int sayi)
{
       Stack<int> s;

       while(sayi)   s.push(sayi--);
       int sonuc =1,temp;

       while(s.pop(temp))   sonuc*=temp;
       return sonuc;
}
```

Buna göre aşağıdaki fonksiyonların yığın kullanılan versiyonlarını yazınız

```
int ustAl(int sayi,int ust)
{
       if(ust==1)    return sayi;
       return sayi*ustAl(sayi,ust-1);
}
```

```
int Fib (int sayi)
{
       if(sayi==1)   return 1;
       if(sayi==0)   return 1;

       return Fib (sayi-1)+Fib (sayi-2);
}
```

```
bool dortgenler[40][40];
//dortgenlere ait değerler daha önceden
atanmış olduğu düşünüleceks
void floodfill(int x,int y)
{
       if(dortgenler[x][y])
              return;
       if(x==0||x==39||y==0||y==39)
              return;
       floodfill(x-1,y);
       floodfill(x+1,y);
       floodfill(x,y+1);
       floodfill(x,y-1);
}
```

```
int Pascal(int x,int y)
{
       if(x==0||y==0||x==y) return 1;
   return Pascal(x-1,y-1)+Pascal(x-1,y);
}
```

```
int F(int x,int y)
{
       if(y==0)
              return x;

       return F(y,x%y);
}
```

```
void Reverse(char* s)
{

       if(*s != '\0')
              Reverse(s+1);
       cout<<*s;
}
```

```
void ReverseNumber(int number)
{
       if(number==0)
              return;

       int mod = number%10;

       cout<<mod;

       ReverseNumber(number/10);
}
```

```
char *Kopyala_r(char *s, char *t, int n)
{
       if (n>0 && (*s = *t) != '\0')
           Kopyala_r(s+1,t+1,n-1);
       return s;
}
```