Class test (#4)

Total points 4/5

Date: October 6, 2020
Maximum marks: 5 (To be normalized to 3)
Expected time to answer 5 questions: 5-7 minutes
Total time: 10 minutes

The respondent's email address (**f20181119@pilani.bits-pilani.ac.in**) was recorded on submission of this form.

0 of 0 points

Name *

Shreyas Kera

ID *

2018A7PS1119P

Questions 1-5 4 of 5 points

Consider a variable declaration as shown in the following code 0/1

```c
#include <stdio.h>
#include<stdlib.h>
int main()
{
    int *p, *q, x;
    x= 25;
    p=&x;
    q=(int *)malloc(sizeof(int)*10);
    return 0;
}
```

- ○ Both variables 'p' and 'q' are allocated space in call stack
- ◉ Variable 'q' is allocated space in heap memory while the variable 'p' is allocated space in call stack
- ○ Variable 'q' is allocated space in call stack memory while the variable 'p' is allocated space in heap
- ○ Both variables 'p' and 'q' are allocated space in the heap memory

Correct answer

- ◉ Both variables 'p' and 'q' are allocated space in call stack

Consider the following code written in C programming language. What is the output of the program? 1/1

```c
#include <stdio.h>
#include<stdlib.h>
int main()
{
    int x, *p, *q;
    x=50;
    p=&x;
    q=(int *) malloc(sizeof(int)*30);
    *q=*p+2;
    x=x+20;
    printf("%d %d\n", *p, *q);
    return 0;
}
```

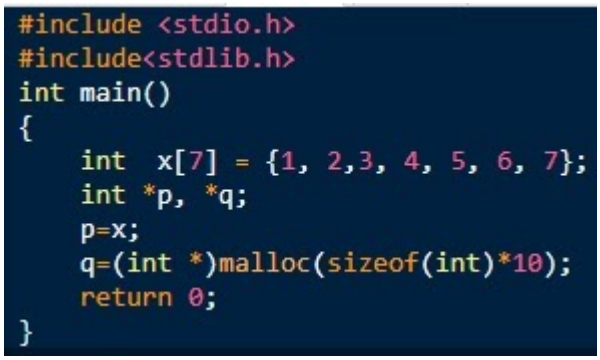
- 52 70
- 52 52
- 20 52
- None of these
- 70 52
- 50 52
- 50 20

Consider the following code. The size of integer is 4 bytes. What is the size of memory leak (in bytes)? 1/1

```c
#include <stdio.h>
#include<stdlib.h>
int main()
{
    int *p, *q, *r;
    p=(int *)malloc(sizeof(int)*45);
    q=(int *) malloc(sizeof(int)*30);
    r=(int *) malloc(sizeof(int)*67);
    *p=23;
    r=p;
    p=q;
    *(q+10)=43;
    r=p;
    return 0;
}
```

- ○ none of these
- ◉ 448
- ○ 180
- ○ 300
- ○ 388
- ○ 120
- ○ 268

Consider the following code written in C language. What is the size of the anonymous variables? 1/1



```c
#include <stdio.h>
#include<stdlib.h>
int main()
{
    int  x[7] = {1, 2,3, 4, 5, 6, 7};
    int *p, *q;
    p=x;
    q=(int *)malloc(sizeof(int)*10);
    return 0;
}
```

- [ ] 2
- [x] 10
- [ ] 1
- [ ] none of these
- [ ] 7

The explicit deallocation of memory by the user can lead to a problem known as 1/1

- [ ] stack overflow
- [ ] memory leak
- [x] dangling pointer
- [ ] none of these