SCE212 Computer Organization and Architecture (Fall 2018)

Student ID: 201721107 Name: [illegible]

Q1. What is the output of the following program?

<- 5 6 7 7

```
#include<stdio.h>

main()
{
   char s1[50], s2[50] = "Hello";

   s1 = s2;
   printf("%s", s1);
}
```

A - Hello

B - No output

(**C**) - Compile error

D - Runtime error

Q2. Suppose that in a C program snippet, followings statements are used.

i) sizeof(int);
ii) sizeof(int*);
iii) sizeof(int**);

Assuming size of pointer is 4 bytes and size of int is also 4 bytes, pick the most correct answer from the given options.

(A) Only i) would compile successfully and it would return size as 4.
((B)) i), ii) and iii) would compile successfully and size of each would be same i.e. 4
(C) i), ii) and iii) would compile successfully but the size of each would be different and would be decided at run time
(D) ii) and iii) would result in compile error but i) would compile and result in size as 4.

Q3. What is the output of the following program?

```
#include <stdio.h>

int main()
{
    int *ptr;
    int x;

    ptr = &x;
    *ptr = 0;

    printf(" x = %dn", x);
    printf(" *ptr = %dn", *ptr);

    *ptr += 5;
    printf(" x  = %dn", x);
    printf(" *ptr = %dn", *ptr);

    (*ptr)++;
    printf(" x = %dn", x);
    printf(" *ptr = %dn", *ptr);

    return 0;
}
```

output:

0
0
5
5
6
6

Q4. What is the output of following program?

```
# include <stdio.h>
void fun(int x)
{
    x = 30;
}

int main()
{
  int y = 20;
  fun(y);
  printf("%d", y);
  return 0;
}
```

output:
20

Q5. What is the output of following program? Assume that an int variable takes 4 bytes and a char variable takes 1 byte

```
#include<stdio.h>
int main()
{
    int arr[] = {10, 20, 30, 40, 50, 60};
    int *ptr1 = arr;
    int *ptr2 = arr + 5;
    printf("Number of elements between two pointer are: %d.",
                                    (ptr2 - ptr1));
    printf("Number of bytes between two pointers are: %d",
                                (char*)ptr2 - (char*) ptr1);
    return 0;
}
```

output: 40
4

Q6. Pick the best statement for the following C program snippet.

```
#include "stdio.h"
int main()
{
 void *pVoid;
 pVoid = (void*)0;
 printf("%lu",sizeof(pVoid));
 return 0;
}
```

(A) Assigning (void *)0 to pVoid isn't correct because memory hasn't been allocated. That's why no compile error but it'll result in run time error.
(B) Assigning (void *)0 to pVoid isn't correct because a hard coded value (here zero i.e. 0) can't assigned to any pointer. That's why it'll result in compile error.
(C) No compile issue and no run time issue. And the size of the void pointer i.e. pVoid would equal to size of int.
(D) sizeof() operator isn't defined for a pointer of void type.

Q7. Choose the best statement with respect to following three program snippets.

```
/*Program Snippet 1 with for loop*/
for (i = 0; i < 10; i++)
{
   /*statement1*/
   continue;
   /*statement2*/
}

/*Program Snippet 2 with while loop*/
i = 0;
while (i < 10)
{
   /*statement1*/
   continue;
   /*statement2*/
   i++;
}

/*Program Snippet 3 with do-while loop*/
i = 0;
do
{
   /*statement1*/
   continue;
   /*statement2*/
   i++;
}while (i < 10);
```

(A) All the loops are equivalent i.e. any of the three can be chosen and they all will perform exactly same.

(B) continue can't be used with all the three loops in C.

(C) After hitting the continue; statement in all the loops, the next expression to be executed would be controlling expression (i.e. i < 10) in all the 3 loops.

(D) None of the above is correct.

Q8. Consider the following function that takes reference to head of a Doubly Linked List as parameter. Assume that a node of doubly linked list has previous pointer as prev and next pointer as next. Assume that reference of head of following doubly linked list is passed to above function

1 2 3 4 5 6.

What should be the modified linked list after the function call?

```
void fun(struct node **head_ref)
{
    struct node *temp = NULL;
    struct node *current = *head_ref;

    while (current !=  NULL)
    {
        temp = current->prev;
        current->prev = current->next;
        current->next = temp;
        current = current->prev;
    }

    if(temp != NULL )
        *head_ref = temp->prev;
}
```

(A) 2 <--> 1 <--> 4 <--> 3 <--> 6 <-->5

(B) 5 <--> 4 <--> 3 <--> 2 <--> 1 <-->6

(C) 6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1

(D) 6 <--> 5 <--> 4 <--> 3 <--> 1 <--> 2

Q9. Following is C like pseudo code of a function that takes a Queue as an argument and uses a stack S to do processing. What does the below function do in general?

```
void fun(Queue *Q)
{
    Stack S;  // Say it creates an empty stack S

    // Run while Q is not empty
    while (!isEmpty(Q))
    {
        // deQueue an item from Q and push the dequeued item to S
        push(&S, deQueue(Q));
    }

    // Run while Stack S is not empty
    while (!isEmpty(&S))
    {
      // Pop an item from S and enqueue the poppped item to Q
      enQueue(Q, pop(&S));
    }
}
```

(A) Removes the last from Q

(B) Keeps the Q same as it was before the call

(C) Makes Q empty

(D) Reverses the Q

Q10. Following is C like pseudo code of a function that takes a number as an argument and uses a stack S to do processing. What does the below function do in general?

```
void fun(int n)
{
    Stack S;  // Say it creates an empty stack S
    while (n > 0)
    {
      // This line pushes the value of n%2 to stack S
      push(&S, n%2);

      n = n/2;
    }

    // Run while Stack S is not empty
    while (!isEmpty(&S))
      printf("%d ", pop(&S)); // pop an element from S and print it
}
```

(A) Prints binary representation of n in reverse order

(B) Prints binary representation of n

(C) Prints the value of Logn

(D) Prints the value of Logn in reverse order